PROCESSING CAPACITY INVESTMENT AND SERVICE PROVISION PROBLEMS FOR TELECOMMUNICATIONS AND THE INTERNET

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Doctor of Philosophy in Management Science

at the

University of Canterbury

by

MARK CHRISTOPHER STEWART

University of Canterbury

New Zealand

2003

ABSTRACT

Given the importance of communication services, such as those provided by telecommunications and the Internet, the problem of designing their networks has been extensively investigated by researchers in Management Science / Operations Research. Particularly, there is much research determining appropriate routing systems. However, given recent improvements in routing technologies and the potential growth in services which require more computing resources, it has become important for service providers to examine their networks from a processing capacity point of view, as these resources are a new potential bottleneck.

This thesis presents models that mathematically describe aspects of processing investment and service provision, given distributed processing in modern communication networks. Unlike previous research, aspects considered include delivery times, quality of service, and congestion. The model developed is different from the standard capacity expansion / facility location models in that the capacity decisions implicitly influence demand.

The mathematical models developed in this thesis for the capacity investment problem contain integer decision variables and uncertain parameters in a two-stage stochastic integer programming framework. These aspects add significant complexity to the

model. Therefore, another important contribution of this thesis is the investigation into the impact of making model approximations to reduce this complexity. This enables better understanding of the model by illustrating the relative importance of the model's complex aspects.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

There are many people without whom this thesis would either have not been possible or would have not been nearly so enjoyable.

Firstly, special thanks go to Dr Shane Dye, whose excellent supervision has always driven this research forward, particularly with his ability to constantly find new questions even when I thought I had all the answers.

I would also like to thank the other members of the Department of Management whom I have come to know over the past few years, particularly my fellow PhD students. You have been a great source of encouragement, and distraction!

To my parents, thank you for getting me to this stage through your encouragement and support, and thank you for always showing a genuine interest in my work.

Finally, thank you for everything Natalie, you have helped more than you probably know.

1. INTRODUCTION

1.1. Motivation

Services that facilitate communication and the smooth transfer of information are a vital component in today's electronic world. Telecommunications services, from simple dial-up telephony to recent inventions like videoconferencing, are now fundamental to society. Internet services, such as electronic mail and search engines, are also rapidly gaining popularity as the growth of the Internet soars (Chapman and Kung (1998) and Sampat (2000)).

The significance of these services, along with recent industry changes detailed in Audestad (1998b) (like deregulation and the resulting increased competition), mean that it is important to investigate how best to meet the ever increasing level of demand. The problem of designing communication networks has been extensively investigated by researchers in Management Science / Operations Research. Particularly, much research has been conducted, using optimisation models and queueing analysis, to determine appropriate routing systems.

Network routing has been targeted mainly because traditional services tend to use more transportation resources (such as switching and bandwidth), and hence these resources were the important network bottlenecks. In recent times, however, some attention has also been given to the computing (or processing) resources required. There are two main reasons for this. First, transportation capacities have increased dramatically recently through the use of better protocols, improved cable types (such as fibre optics), computer-based switching, and data compression technology. Second, like Tomasgard (1998), we believe that the future growth in demand will be for services requiring more computing resources, including search engines and multi-media applications like videoconferencing, video on demand, and real-time web streams. Therefore, whilst recognising that transportation bottlenecks will never be completely eliminated, it has become important for service providers to examine the networks from a processing capacity point of view, as these resources are a new potential bottleneck.

While it is becoming important for service providers to expand their processing resources, there is little research into the best way for them to do this. Kreidi and Sanso (1994) is one example, where workload distribution in a network of computers is investigated with whole jobs being processed on a single machine. However, given recent technological developments, such as digital technology and faster transportation networks, the resulting advent of distributed processing environments (for example, Barr *et al* (1993)) means the processing of jobs can be distributed over the network. As far as we are aware, Tomasgard (1998) was the first to look at optimisation models considering processing-based services in such an environment. This research, and its more recent extensions, does not consider aspects of network transportation. However, because services (video on demand and real-time web streams, for instance) can have high transportation, as well as high processing requirements, it is necessary to consider the transportation aspect (and the network congestion it can cause) when expanding processing resources.

1.2. Problem Description

This section provides a background for this thesis, setting the framework for our view on capacity expansion, service provision, distributed networks, and network design.

1.2.1. Capacity Investment and Operational Decisions

Capacity investment, or *capacity expansion, decisions* recognise the importance of adequate capacity levels by setting the service provider's network resources (processing and routing). Given fixed resources, *operational decisions*, or *processing decisions*, determine how to meet demand. As highlighted by Karmarkar and Kekre (1987), the capacity investment decisions should be made considering the environment and decisions they impact, namely the operational decisions. The way demand is met at the operational level will influence the best decisions for investments in the processing infrastructure. Hence, whilst modelling the capacity investment decisions is the main aim of this thesis, in order to do this we must also model the processing decisions.

It is important to consider the time frame that both these sets of decisions are made over. The capacity decisions are likely to be made considering a planning horizon of a number of years. Conversely, the length of the processing decisions period is likely to be relatively short, possibly in hours or days. Hence, as shown in Figure 1.1, the capacity decisions set the resources for many processing decisions periods. Over the long time frame for the capacity decisions, uncertainty will be introduced. Because an infinite planning horizon would make the problem intractable, we use a finite planning horizon. Lundin and Morton (1975), for example, look at how to choose the length of this planning horizon to ensure a stable solution.

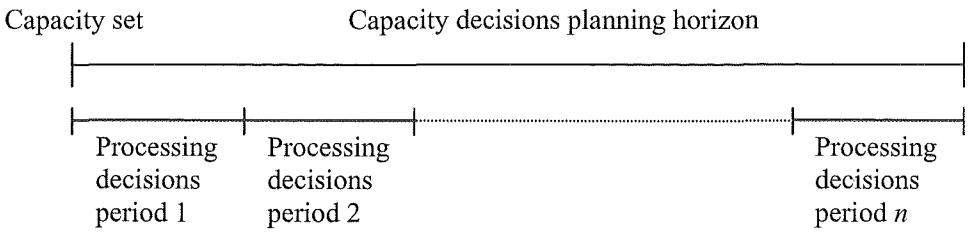


Figure 1.1. Planning horizons of the capacity investment and processing decisions.

1.2.2. Distributed Processing Environment

Distributed processing aspects are crucial to the modelling in this thesis. Audestad (1998a) provides a general overview of distributed processing, while Mullender (1993) details the technological side of distributed processing in networks. An example of a future distributed processing standard has been defined by the Telecommunications Information Networking Architecture Consortium (TINA-C) (see, for example, Barr *et al* (1993) or Inoue *et al* (1998)). TINA is designed with an integrated service approach in mind, promoting reuse of the software for services and hiding the complexities of its underlying technologies from the service designers and users (Kitson *et al* (1994)). It also assumes that service components (parts of a service) can be met at different places in the distributed network. This increases flexibility in resource allocation.

Here, we give a brief description of the assumed properties of the distributed environment as used in this thesis. A distributed processing environment is composed of services, the computing nodes that process the services, and the underlying transportation networks that route flow stemming from processing services. Figure 1.2 (from Tomasgard (1998)) describes the relationship between these components. The processing nodes, where the applications to produce services reside, join the distributed services architecture and the underlying network architecture together. The solid lines in the upper level of this figure represent how applications interact to produce services. This interaction creates flows between the processing nodes on the underlying networks. Different processing nodes can be used to provide different

applications, an interaction shown by the solid lines in the middle level of the figure. However, the real transportation required to do this is performed by the underlying networks (the lower level).

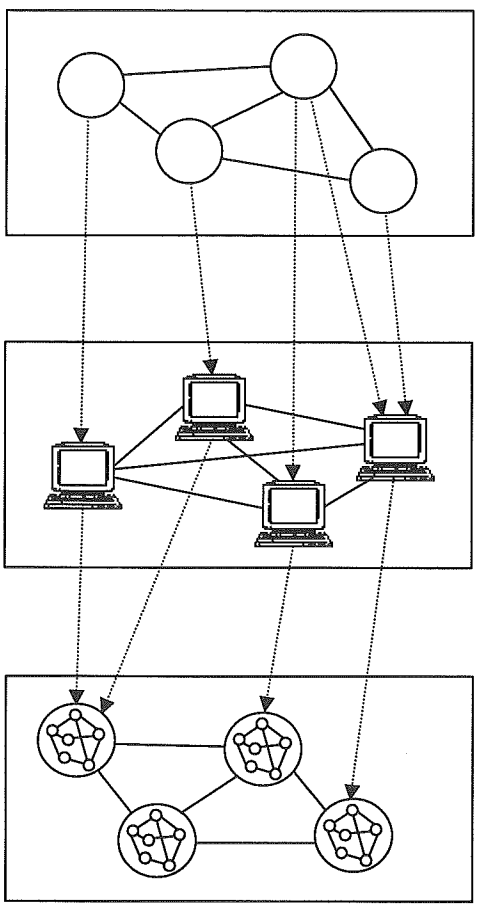


Figure 1.2. The relationship between transportation networks, processing nodes, and applications.

1.2.3. Service Provision

This section describes the service provision framework within which this thesis is set. Terminology used in this research is *italicised.* The section finishes with a diagrammatical overview, Figure 1.4.

1. *Service providers provide end services to customers; network providers provide service providers with their connection to the underlying routing network.*

Service providers are companies that offer services to the customer. Examples of *services* include videoconferencing, search engines, and electronic mail. We assume that strategic decisions, such as what services to provide, have been made. *Network providers* operate the communications systems that route network flow, and hence are the service provider's connection to the underlying routing network. Either service providers or network providers can own the processing nodes. This means service providers can exist without owning their own network hardware. Referring back to Figure 1.2, this means that the service providers control the top two layers of this diagram, whilst using the lower layer to aid in service provision.

2. *Demand for the different services can be received at a finite number of potential demand locations.*

A service provider can receive demand for many types of services from potentially anywhere in the world. Because customers who request services from the same geographical vicinity will have similar transportation requirements, these customers are grouped to form a finite number of *demand locations*. Hence, a service provider views its demand as being for different services from a finite number of different locations in the overall network. Note that the models developed in this research do not restrict individual customers from being considered, but the resulting model size does.

3. *Demand for a service leads to independent demands for the subservices that make up that service.*

A communications service consists of a number of different parts. For example, a videoconferencing service has, among other things, video and audio components. To utilise the inherent flexibility of the distributed processing environment, services are broken into these smaller entities, called *applications* or *subservices*. A subservice is defined as being the smallest entity from which services can be constructed. A service is usually built from many subservices. In order for the service to be met,

these subservices are processed in the network separately, as determined by the service provider. Minimal communication is required to co-ordinate the subservices to produce the service requested. For a more detailed explanation of the use of subservices in a distributed processing environment see Dye *et al* (1998). As supported by TINA-C, we assume that subservices which together constitute a service may be distributed on several processing nodes in the network without affecting the interaction between them.

Subservices are used to take advantage of the flexibility and increased efficiency of a distributed processing environment. The same subservice can be shared between services, reducing duplication and saving processing capacity. A subservice can be present at more than one location in the network. Note that the use of services that are not distributed is not precluded by the model, as these services could be viewed as consisting of only one subservice.

4. *Demand requests can be rejected or met.*

Because of constrained resources, or for profitability reasons, the service provider might *reject* a customer's service request. Rejecting demand is when the customer receives a busy signal (Simampo and Ryan (2001)). The customer may or may not make the same request later.

5. *A demand request must be routed and processed in order to be met, incurring a delivery time.*

A service provider meeting a service request consists of two parts: routing and processing. Because both these aspects take time, there is a delay, or a *delivery time*, associated with meeting requests.

6. *Processing of subservices is performed at processing nodes, using the processing capacity installed on those nodes. However, the software required to run the subservices must be installed before this can be done. Congestion can occur at processing nodes.*

Services considered in this thesis require significant network processing. Processing is the actual meeting of demand, the satisfying of the service request. Processing of

subservices is performed using a *processing node* (computer), which has limited *processing resources* or *processing capacity* (memory, for instance). Processing nodes can be installed at any of a finite number of potential locations in the network, and are designed so that multiple programs (subservices) can be run on them.

To enable a processing node to process demand for a subservice, the software required to run that subservice must first be installed; this takes a fixed amount of processing capacity. Once a subservice is installed, multiple requests for that subservice can be processed at the node. Hence, processing capacity is used in two ways: as a fixed requirement to make a subservice available on a processing node, and to actually meet demand at a processing node. The more processing capacity used at a processing node the more congested the processing node will become, slowing the processing rate (see Roberts (2001) for a good explanation of *processing congestion*). As it uses processing capacity, installing subservices adds to this congestion. Note that, because of the congestion resulting, it may not be in the service provider's best interests to use all their capacity.

7. *The flow resulting from a demand request must be routed over arcs and through routing nodes. Congestion can occur at routing nodes.*

A service provider may not always meet requests at a customer's closest processing node (for example, to avoid processing congestion at that node). When this happens, *flow* relating to these service requests must be routed over the underlying network for the processing to be performed at the chosen node. The major flow (in terms of volume) is between the nodes which process a request's subservices and the customer's location. Other, more minor, flows, such as the customer's request, rejection notification, or flows required to determine where to process demand, are not considered in the model. Also, transportation from the nominated centre of a grouped demand location to the actual customer is assumed to not add to congestion.

Flow is routed between nodes using the international structure of telephone and data networks under a protocol like TCP / IP (see Cleveland and Sun (2000) for a simple introduction to TCP / IP and other protocols, or Miller (1997) for more detail). These networks consist of routing nodes and arcs. *Routing nodes* refer to the routers and

switches that forward flow through the network, avoiding congestion for better response times. They do this using routing table software that provides them with information on how busy the network ahead of them is (Dowd (1997)). Congestion can occur at routing nodes as more flow waits to be routed through the network. While routing nodes support the software that determines how to route flow, the *arcs* form the network component that actually moves the flow. Depending on the region they could be anything from copper cables operating in circuit-switched networks, to fibre optics operating in an integrated services digital network (ISDN) using frame relay or asynchronous transfer mode (ATM) technology. The reader is referred to Stallings (1995) for a review of this technology. Arcs have a certain capacity (bandwidth), and carry flow at a constant rate until that capacity is reached. For a more detailed explanation of these network components, see Miller (1997) in particular.

Hence, a route is defined by its starting and ending locations, and the arcs and routing nodes used. There can be many different routes between the same locations, and routes between different locations can share common arcs and nodes. If the demand is processed at the location at which it was received then it is assumed no routing of flow is required.

In this thesis only an overview of the complicated routing network is used. For example, an arc in the network could actually represent a combination of arcs and routing nodes. This is illustrated in Figure 1.3. A simplified network was used because it is important to consider routing, but we do not want routing to 'over-power' the model.

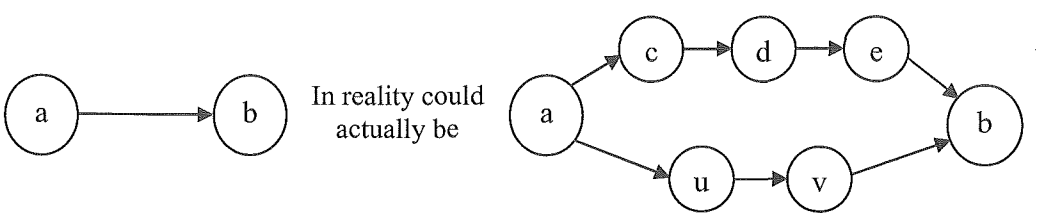


Figure 1.3. Arc approximation.

8. *The database component is automatically stored at a location when the subservice is installed, not adding to congestion.*

Preparing a processing node for processing of subservices consists of two parts: obtaining the necessary software to ensure the service can be run there, and ensuring that the necessary data (i.e., the accompanying database) is available. The database would be installed at the node (for instance, on the hard drive of the computer), and does not add to processing congestion. Considering external databases is a straightforward extension that this research excludes.

9. *Diagrammatic overview*

Figure 1.4 shows a simple overview of the decision-making environment for this research, where the 'world' is being approximated by a network of demand and processing locations. A node in the network may receive demand, process it, or both. Routing nodes exist throughout the network, on routes between locations, and most of these are not shown in Figure 1.4. All processing and demand locations are also routing nodes. Considering the routing network is important because, although we do not consider routing network expansion in this thesis, routing ability still substantially affects the processing capacity expansion decision. For example, the service provider would not want to install processing nodes in areas of high congestion.

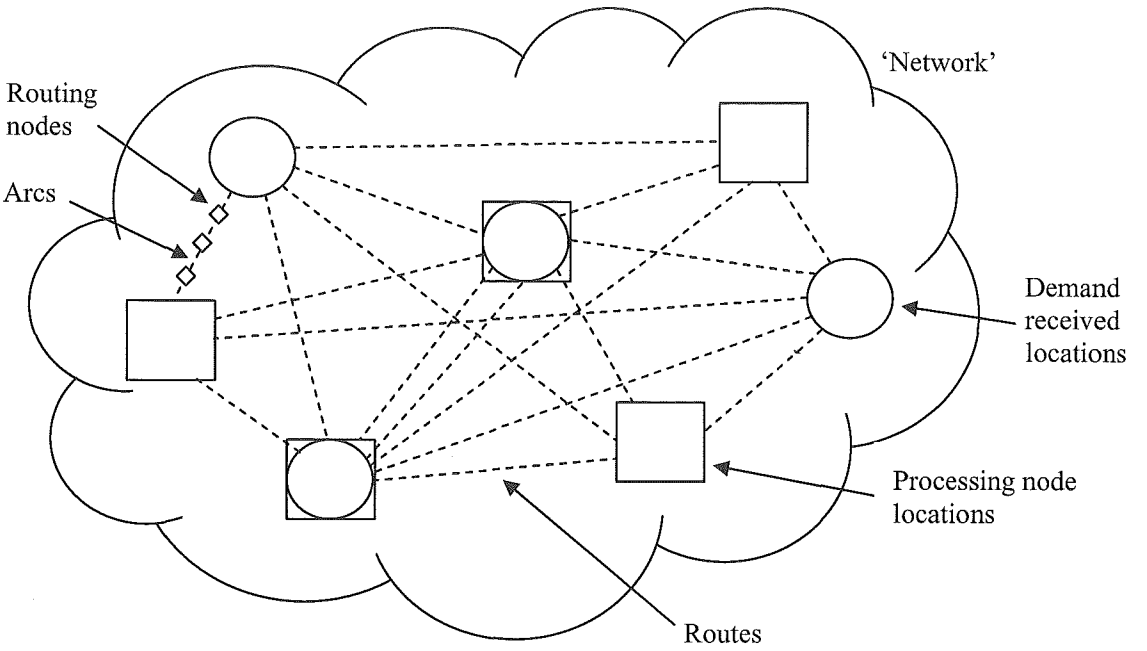


Figure 1.4. Modelling the receiving, routing, and processing of demand.

1.3. Research Contribution

As far as we are aware, Tomasgard (1998) was the first to develop optimisation models that capture the flexibility of a distributed processing environment and the characteristics of processing-based communication services. This research looked, in detail, at the operational decisions for providing services in this environment, as well as beginning to investigate the strategic processing investment problem. Tomasgard's research assumes that transportation resources are not a network bottleneck, and subsequently all factors associated with this are excluded from consideration. However, in many modern networks it is important to consider the routing system and its associated aspects because congestion plays such an important part.

Therefore, one of the main contributions of this thesis is the research that mathematically describes aspects of processing investment and service provision, given distributed processing in modern communication (telecommunications / Internet) networks. Unlike previous research, aspects include delivery times, quality of service, and congestion; considering these aspects makes the problem considerably

more complex. This problem is different from a standard capacity expansion / facility location problem (for example, Manne (1967)) in that, by considering the aforementioned aspects, the capacity decisions implicitly influence demand. This is because the capacity levels influence the quality of service that the service provider is able to provide, which in turn can influence future demand. To our knowledge this research is the first to provide an in-depth discussion modelling and analysing capacity expansion decisions where demand is endogenous, dependent on capacity and the resulting possible service levels.

As discussed in Chapter 2, there is much existing literature on design of the underlying routing network. As such, this aspect is not considered in this research; routing locations and capacities are assumed known. Hence, the capacity expansion decisions in this thesis are viewed as being made by either a current service provider, who has existing routing networks, or a provider who intends to lease routing facilities.

The changes to the communication industries, particularly deregulation and technological improvements, have meant competition has increased and companies have had to start adapting to this change (see Fowler and Wright (1994) for an overview of these changes). For example, as discussed in Messerschmitt (1996), the media that industries traditionally dealt with (audio and video for telecommunications and data for computing) are being integrated together to produce multimedia applications. This has blurred the difference between telecommunications companies and Internet service providers to the extent that the models in this thesis are suitable for both industries.

The mathematical models developed in this thesis for the capacity investment problem contain integer decision variables and uncertain parameters in a two-stage stochastic integer programming framework. These aspects add significant complexity to the model. Therefore, another important contribution of this thesis is the investigation into the impact of making model approximations to reduce this complexity. This enables better understanding of the model by illustrating the importance of the respective model parts.

Note that while computational results will be presented for the models developed in this thesis, we do not intend to model a particular service provider's problem, or provide empirical evidence of optimal strategies for service providers. Rather we develop models as generically as possible, so as to make them relevant for more users. However, to the best of our ability, we attempt to use realistic data in our test problems.

1.4. Outline

As detailed in Section 1.3, the main contribution of this thesis is the development of a capacity expansion model that endogenously models demand, dependent on capacity and service levels. Chapters 3, 5, and 6 directly model the system of interest, culminating in the presentation of the capacity expansion model in Chapter 8. Specifically, Chapter 3 models the processing decisions environment given fixed capacity. This is important because how demand is met at the operational level will influence the capacity expansion decisions. It is important to note, however, that whilst the model developed in this chapter forms the basis for the operational model used in the capacity expansion model, there are important subtle modelling differences. These differences, as well as other considerations in the strategic environment, are outlined in Chapters 5 and 6.

Chapter 4 looks at the properties of the model developed in Chapter 3, including obtaining information that is useful for the testing in Chapter 11. Chapter 7 models demand changes explicitly over the planning horizon, but for a fixed capacity decision. In comparison, the model presented in Chapter 8 approximates demand evolution by modelling long-run demand, but allows for optimisation of the capacity expansion decision. The model developed in Chapter 7 is also used as a simulation in Chapter 11 to estimate expected long-run profits of capacity solutions.

Chapters 9 and 11 add to the understanding of the model developed in Chapter 8. Chapter 9 investigates the solution process of the capacity expansion model. Chapter 11 explores aspects of the model that contribute to the model complexity, analysing the impact of making model approximations to reduce this complexity.

In summary, the remainder of this thesis is organised as follows:

- Chapter 2 outlines the literature regarding capacity expansion decisions, particularly those most pertinent to this thesis; in routing and processing resource expansion in communications networks.
- Chapter 3 describes the processing decisions in greater detail, discussing the framework for the decision problem and the different aspects considered. A mathematical model for the single-period problem is developed.
- Chapter 4 presents analysis of this processing decisions model, particularly investigating what effect approximations to the model have on the types of solutions the model produces. This adds insight into the applicability of the operational model approximations outlined in Chapter 11.
- Chapter 5 describes in greater detail the environment in which the processing capacity investment decisions are made, including the adaptations to the processing decisions model necessary for use in this longer planning horizon.
- Chapter 6 discusses one of the most important, and complex, factors in the system of interest for the processing capacity investment decisions: demand. This includes discussing the important interaction between capacity, congestion, quality of service, and demand.
- Chapter 7 outlines a model developed to explicitly model demand evolution given a fixed capacity decision. This model is later used to simulate how demand changes from period to period.
- Chapter 8 presents the two-stage stochastic integer mathematical model developed for the processing capacity investment decisions. Stochasticity of parameters is also discussed.
- Using a simplified deterministic example, Chapter 9 aids understanding of potential solutions produced by the model presented in Chapter 8. The impact of stochasticity on solutions is also investigated.
- Chapter 10 outlines possible extensions to the capacity model.

- Chapter 11 presents computational results from the capacity model, concentrating on illustrating the impact of approximating aspects of model complexity.
- Chapter 12 draws conclusions and proposes areas for future research.

2. LITERATURE REVIEW

This chapter reviews the capacity expansion literature, identifying the need for more research on the problem of expanding network processing capacity while considering transportation and quality of service issues.

2.1. Introduction

Because of its importance, much literature exists discussing, modelling, and solving the capacity expansion problem. This chapter reviews this literature. Section 2.2 provides an overview of general capacity expansion problems. This overview will focus on model development. Section 2.3 goes into more detail, reviewing capacity expansion problems particularly germane to this thesis. An overview of the research conducted in the Internet, particularly in Internet routing systems, is provided in Section 2.4. Finally, Section 2.5 provides a thorough review of research on the

problem of expanding network processing capacity, for which there is little existing literature.

2.2. General Capacity Expansion Problems

Management Science / Operations Research has been used since the late 1950s to develop models and solution approaches suitable for many capacity expansion applications. As a starting point for our review of the general capacity expansion literature we refer the reader to Luss (1982). This survey paper organises the capacity expansion literature up to that date, establishing a framework for capacity expansion problems from an Operations Research perspective.

As a reference point, the model presented in Figure 2.1 is a classic multi-facility capacity expansion problem, similar to that presented in Luss (1982). The objective (1) minimises capacity expansion and transportation costs. Constraint (2) ensures that the capacity at location n is not exceeded, and Constraint (3) ensures demand is met. Other papers extend the model complexity and / or solution techniques.

Definitions:

dr = location where demand is received.

n = potential processing location.

$C_n(\,)$ = the total investment cost function for location n.

d_{dr} = demand received at dr.

$t_{dr,n}$ = transportation costs between demand location dr and processing location n.

G_n = units of processing capacity added to location n.

$X_{dr,n}$ = demand received at dr that is met at location n.

Formulation:

$$\textit{Min} \qquad \sum_{n} C_n(G_n) + \sum_{dr} \sum_{n} t_{dr,n} X_{dr,n} \qquad (1)$$

s.t. Constraints

$$X_{dr,n} \leq G_n \qquad \forall\, dr,n, \quad (2)$$

$$\sum_{n} X_{dr,n} = d_{dr} \qquad \forall\, dr, \quad (3)$$

$$G_n, X_{dr,n} \geq 0.$$

Figure 2.1. Classic capacity expansion formulation (Luss (1982)).

2.2.1. Luss (1982): Capacity Expansion Literature Survey

According to Luss (1982), "the basic capacity expansion problem consists of determining the sizes of facilities to be added and the associated times at which they should be added". In addition, where transportation costs between locations are involved, the appropriate location for any expansion is important. Where applicable, the type of capacity installed is also important. Expansion size, G_n, is predominantly a continuous variable.

Aspects to Consider in Capacity Expansion

Luss discusses the issues most prevalently considered in the literature in capacity expansion problems, and the model complexity they can cause.

Capacity expansion costs generally exhibit substantial economies of scale: the average cost per capacity unit decreases with the expansion size. This introduces an important trade-off into the model, between economies of scale savings of large expansion sizes and the cost of installing capacity before it is needed. The expansion cost function, $C_n(G_n)$, is usually concave, such as the fixed charge function shown below, or piece-wise concave if different technologies are present.

$$C_n(G_n) = \begin{cases} 0 & \text{if } G_n > 0 \\ A + B\ G_n & \text{if } G_n = 0 \end{cases}$$

Demand (d_{dr}) is an important issue in the model because capacity is set depending on it (constraint (3)), and because the way demand is modelled significantly affects the complexity of the model, and hence the appropriate solution technique. Demand is represented in the literature by many different functions. The simplest is a linear function with a constant demand growth rate over time. Exponential functions are also used, where demand growth increases, proportional to demand volume at a point in time. Finally, demand often has a decreasing growth rate over time, with an asymptotic maximum level. In other cases demand over time is not fully known. Treating the demand function as a stochastic process helps to examine the effect of demand uncertainty on expansion policy.

Other important issues in the capacity expansion decision include allowing for the possibility of shortages, where some demand is temporarily unsatisfied. In a production environment, inventory can be used to cover this. Congestion costs are relevant in applications where high capacity utilisation causes congestion. Also, often there is a facility maintenance cost associated with excess capacity. The discount rate also has a significant impact on the optimal expansion policy. Note that operating costs are generally assumed to depend only on demand volume, not capacity, and as such are constant. Luss notes that, for this reason, they are excluded from most models.

The capacity expansion problem's objective function (for example, (1)) is typically to minimise the discounted costs associated with the expansion process. This includes, where appropriate, costs for expansions, shortages, congestion, idle capacity, maintenance, and inventory. Possible constraints imposed, other than capacity restrictions (constraint (2)), include budgetary limitations and acceptable policy plans (such as upper bounds on expansion sizes, excess capacity, and capacity shortages).

Types of Capacity Expansion Problems

Luss (1982) distinguishes between papers that investigate single or multi-facility problems using an infinite or finite planning horizon. For the single-facility expansion problem solution techniques used include dynamic programming, shortest path heuristics, and branch-and-bound algorithms. Modifications to solution algorithms to consider model extensions are discussed. For the multi-facility expansion problem, a new dimension to the expansion decision, location, is added. This is because capacity at a location can be used to satisfy other locations' demand, but a transportation cost ($t_{dr,n}$) is incurred. Initial heuristic solution procedures have been developed for the multi-facility problem.

Capacity expansion planning using an infinite planning horizon is complicated. For finite period problems the time scale in the model is represented by discrete time periods t = 1, 2,..., T, where T is the length of the finite planning horizon. The special characteristics of this finite horizon problem aid solution processes significantly.

Mathematical models reviewed for these types of capacity expansion problems range from simple models with linear deterministic demand to more complex models that have more than one capacity type with stochastic demand. See Luss (1982) for an extensive list of references.

Capacity Expansion Application Areas

After the emphasis on modelling approaches and algorithmic solutions, Luss provides brief descriptions of capacity expansion problems in many application areas. These areas include water distribution, sewage disposal, road transportation, electricity transmission networks, manufacturing systems, heavy process industries (aluminium, chemicals), and, of course, communication networks. A summary of relevant references is provided for each application area.

Seminal Research

Whilst we have not, and will not, go into detail about papers before Luss' excellent review, we will mention some of the seminal references. In one of the earliest examples, Manne (1961) outlines simple models with deterministic demand that grows linearly over time. Capacity is expanded by x units whenever demand reaches capacity, where the expansion size depends on the economies of scale of large expansions and the discount rate. Secondly, citing various authors, Manne (1967) describes a multitude of capacity expansion problems, both in applied and theoretical contexts. Of particular note, Erlenkotter (1967) introduces the multi-facility expansion problem. The first well-known case study on the application of capacity expansion problems is that described by Manne (1967) for heavy process industries in India. As a final reference, Freidenfelds (1981) is a more recent resource for understanding variations of the capacity expansion problem.

2.2.2. Recent Research

Since Luss's review of the literature much research has been conducted in this problem area, particularly on the multi-facility problem. There have been significant extensions in solution techniques for multi-facility problems, an area of weakness identified by Luss (1982). Sridharan (1995) provides a recent review of the various solution methods, heuristic and exact, for the capacitated plant location problem. Extensions in model development have centred around considering different technologies, capacity deterioration, demand stochasticity, congestion and service levels, and on combining the capacity replacement and expansion problems. This section will provide an overview of these developments, all of which are pertinent to this thesis.

Stochasticity

Deterministic capacity expansion models abound in the literature, and will continue to be developed and solved as stepping stones to the more realistic stochastic models. Examples of models with deterministic demand include Erlenkotter (1973),

Freidenfelds (1981), Rajagopalan (1992), Li and Tirupati (1994), and Rajagolapan (1994). However, Laguna (1998) states that "the importance of considering uncertainty in capacity planning is well-known".

Manne (1961) was one of the first to consider stochasticity in capacity expansion models, concluding that it was possible to consider stochasticity using an equivalent deterministic problem by discounting all costs with a lower rate. Using a scenario-based approach to capture demand uncertainty, Eppen *et al* (1989) consider a stochastic mixed-integer program with recourse, paying particular attention to the trade-offs between risk and returns associated with investments in capacity. They also consider the impact of rejecting demand. Chen *et al* (2002) adds to the relatively sparse literature in the stochastic area by developing a stochastic programming model for determining capacity and technology decisions in a manufacturing environment. Solution algorithms are developed to solve this complex problem. They also provide a brief summary of the stochastic capacity expansion literature. Ierapetritou and Pistikopoulos (1994) and Laguna (1998) also look at solution techniques for the stochastic problem. Other examples of papers with two-stage models, where the first stage sets the strategic capacity decisions, and the second stage the operational capacity allocation decisions, include Bienstock and Shapiro (1988), Fine and Freund (1990), and Sen *et al* (1992).

Contraction, Deterioration, and Replacement

Aneja and Chaouch (1993) and Rajagopalan and Soteriou (1994) consider models where capacity can be either expanded or contracted over time. Dynamic programming is used to determine the optimal policy. This type of formulation has fast solution times for problems with few facilities, but requires discrete expansion sizes; linear and mixed integer formulations do not have such restrictions. Rajagopalan (1992) looks at decision-making when there is capacity deterioration over time, in that the operating costs are a function of the age of the facility and the extent of utilisation of capacity. This is an important factor in multi-period capacity expansion models.

Because of the inefficiencies and high operating costs of old facilities, incorporating the availability of new technology to replace existing capacity is often desirable. Because of economies of scale in purchasing new machines, it can be cheaper to combine replacement and expansion decisions. Recent research has combined these strands of the literature (for example, Chand *et al* (2000), Rajagopalan (1998), and Rajagopalan *et al* (1998)).

Different Technologies

A number of recent papers consider investing in different technologies (for example, Chen *et al* (2002) and Dasci and Verter (2001)). These papers particularly deal with flexible technology (which can meet any demand) versus dedicated technology (which can meet demand for only one product / service). Rajagopalan (1994) investigates capacity expansion of different technology types in a deterministic environment. This is an important decision in industries where technology is rapidly improving and more than one type of technology is available.

Service Industries

Berman and Ganz (1994) recognised that little research had been published on the capacity expansion problem specifically for the service industry. This is important as capacity expansion in the service industry has a number of different problem aspects to that of the manufacturing environment, particularly not being able to use idle capacity to meet future demand (inventory). They define the service capacity problem as finding a schedule of capacity expansions for each location to maximise profit while satisfying demand (deterministic) in each of a number of markets in all periods over a finite time horizon. Limited funds are available for capacity investments, and demand can only be met where and when it is received (such as is applicable for restaurants and hotels). This problem is modelled as a linear program (with linear investment costs) and as an integer program with a fixed charge for capacity expansion. Solution algorithms are found for both cases.

Congestion Costs and Service Levels

Of particular importance to this thesis, a service provider's capacity influences the level of service they can provide. It is important to consider service ability when making capacity expansion decisions, since poor customer service can have serious negative impacts for a service provider. Li and Tirupati (1995) were one of the first to consider customer service levels in their capacity expansion model. They focus only on whether demand is met or not. Also of importance are the service lead times. Early models (for example, Luss (1982)) have a constant unit cost of transportation between facilities, which could be thought of as including the costs of lead times. However, in many environments this is inappropriate, as congestion slows lead times as capacity utilisation increases, and hence there are increasing marginal costs. Buss *et al* (1994) add to the growing literature that argues capacity utilisation rates of 100% are not suitable, because of the associated undesirable congestion. Luss (1982) notes:

> *"Congestion costs may be incurred even if the capacity exceeds demand. For example, in communication networks a certain amount of spare capacity of transmission and switching facilities is needed in order to manage the network efficiently [to avoid congestion]."*

Karmarkar and Kekre (1987) show that capacity and configuration decisions should be made considering operational costs, including the costs of long lead times, costs of congestion, and shortage costs. In a manufacturing setting, Rajagopalan and Yu (2001) add a constraint to the model to ensure that target delivery times are met with a pre-specified probability. Delivery times and the impact of congestion are modelled using queueing formulae, and are hence non-linear; they therefore derive linear simplifications. Demand that is not met (as it would cause delivery times to exceed targets) is simply lost. So and Song (1998) develop models that attempt to provide an understanding of the interrelationship between simultaneous pricing, delivery time guarantees, and capacity expansion decisions. Delivery times are again included using queueing formulae. Demand is assumed known for a given price and delivery time guarantee.

Recent literature has begun to recognise the link between demand and capacity. This link exists because capacity influences service levels, which in turn influence demand. Klassen and Rohleder (2001) discuss combining demand and capacity management, where "demand management is an attempt to shift demand, while capacity management is a response to demand", without developing mathematical models for the problem. Roberts (2001) also recognises the link between demand and capacity, but again does not model it. Buss *et al* (1994) show that making capacity and demand decisions together improves decision-making. They consider demand as being variable, but it is set by price and advertising, not by service levels. In Chapter 6 we provide the first attempt that we are aware of at modelling demand as variable, being influenced by service levels.

Real Options

In recent times, the use of one particular class of financial derivatives, namely options, has been extended in the literature to investment projects where flexibility plays an important role. Because this is applicable to capacity expansion projects it is important to briefly outline this body of literature.

In capacity expansion projects the decision maker could invest immediately and expand capacity. Alternatively, they might wait a year before deciding whether to expand capacity, in anticipation of better information. The value of this "option" is that the decision maker may know more in a year's time about the uncertainty under which the capacity expansion decisions are made. If conditions are favourable in a year the expansion can go ahead, but if conditions are unfavourable no expansion is required and the losses that would have been incurred by an expansion can be avoided.

Having formulated the decision problem as a financial option, we can value the option using traditional option valuation techniques. In many situations, the flexibility inherent in the decision process can offer additional value that cannot be incorporated into traditional discounted cashflow valuation techniques. Dixit and Pindyck (1994) is a seminal reference for the real options literature, and numerous variations, and a

range of applications, have been developed since (for example, Smith and McCardle (1999)).

2.3. Capacity Expansion in Telecommunications Networks

There is much research conducted in capacity expansion problems in telecommunications networks. This section will review this literature, which, as discussed previously, has mostly focused on designing the underlying routing network. We focus on whether the research considers single or multiple expansions over the planning horizon (referred to as the single-period and multi-period problems), as this feature is the most common differentiating point among models. Also, because it is of particular importance for this thesis, we review how the literature has treated one particular aspect of the network expansion problem: service delay. Because the capacity expansion problem is a reasonably standard, but complex, problem, a significant number of the extensions in the literature over time have focused on improved solution methods. We briefly review this aspect.

Seminal Research

Christofides and Brooker (1974) was an initial work considering capacity expansion in telecommunications networks. They determine which arcs to add to an existing network to maximise flow from source to sink nodes, subject to budgetary limitations. Another early work, Boorstyn and Frank (1977), look at the backbone network design problem, particularly that of locating terminals. Fratta *et al* (1973) present the classic flow-deviation algorithm, a solution method for network design problems. Gavish (1992) provides a summary on the evolution of communication network technologies and design. We recommend this paper, and the references therein, as an excellent place to start for literature on the routing expansion problem.

Single-period Problem

The single-period capacity expansion problem involves simultaneously setting the network capacities and the (fixed) routes to be used in the network. Often (see for example, Gavish and Neuman (1989)) these decisions are made to ensure acceptable

performance at a minimum cost. Costs of increasing capacity are traded-off with costs of delay. Whilst not developing mathematical models, Balakrishnan *et al* (1991) provide a useful overview of this problem, including an outline of characteristics of telecommunication networks, and a discussion of possible modelling approaches in times of improving technology (such as ISDN). They also propose methods for solving the single-period problem. For an applied example, Klincewicz (1994) outlines the optimisation models that AT&T Bell Laboratories have developed for their network design problems.

Gerla (1973) presents simulation results suggesting that at *steady state* there is no significant difference between the delays induced in a network by good static and adaptive routing strategies. Static routing has fixed routes, whereas adaptive, or dynamic, routes change depending on the short term variations in network traffic. Because of these findings, most of the literature uses static routing in the design process. Of course, dynamic routing is favourable for the period-to-period routing at the operational level (because each period's routes are not at 'steady state' in this case). Gavish (1992) provides references for many studies that use static or adaptive routing strategies.

Solution methods for this problem are varied. Gerla and Kleinrock (1977) develop heuristics that solve this problem by iterating between the strategic capacity and operational routing decisions until a good solution is found. Other heuristic solution techniques use different methods. Gavish (1982) looks at formulating the problem as a minimal spanning tree problem. Associated solution procedures are developed. Gavish and Neuman (1989) simultaneously solve the two-stage integer problem, using Lagrangean relaxation and subgradient optimisation techniques to obtain good solutions and tight lower bounds. Flippo *et al* (2000) present a dynamic programming algorithm for the single-period problem. Agarwal (2002) develops a heuristic algorithm for solving the network design problem with several facilities of different capacities and costs. Finally, Lee *et al* (2001) provide guaranteed optimality for the ATM switch capacity allocation problem.

Multi-period Problem

The multi-period problem differs from the single-period problem in that, as well as considering other aspects, the trade-off between economies of scale savings in investment and the cost of maintaining excess capacity must also be considered. Single-period models cannot be directly used for each period in the multi-period problem because this does not allow this trade-off to be considered. The multi-period model and its dynamics were investigated initially by Minoux (1987). Dutta and Lim (1992) allowed for discrete capacity choices. They report excess capacity may result from their linear approximation of the non-linear network performance calculation. Chang and Gavish (1993) formulate models for the network topology design and capacity expansion problem. Chang and Gavish (1995) extend this by developing tighter formulations and new solution procedures. Cox (1997) formulates a complicated model for expanding terminals and links, but does not consider delay costs in his formulation, just expansion costs. Garcia *et al* (1998) look at solution algorithms for network capacity expansion models where traffic requirements are dynamic, but they do not consider congestion. Amiri and Pirkul (1999) study the communication network design problem considering the variation of traffic over a day. This is important because a network designed using average traffic may fail to accommodate demand during peak times.

Dutta and Kim (1996) develop a heuristic for the multi-period capacity expansion problem, removing some of the simplifications of previous heuristics (particularly concavity restrictions on the expansion cost function). Jack *et al* (1992) outline their software system that is used by hundreds of local area planners in a major US telecommunications company to solve their multi-period network expansion problem. They solve this complex problem by decomposing it into two smaller subproblems (the strategic and operational levels). Le Blanc *et al* (1999) provide a review of the solution methods previously employed in the literature.

Service Delay

Service delays, caused by poor network performance, lose customers, and reduce long-run profits. This has been recognised in the literature, as commented by Amiri *et al* (1999) below, and it is important to include this aspect in the capacity expansion model.

> *"Trade-offs have to be made between revenue maximisation and response time to users. If the revenue maximisation factor alone is considered, network users will experience significant delays, and the quality of service will deteriorate."*

Network performance is traditionally measured by average packet delay. Upon recognising that queues and network congestion form as flow increases, the literature in general models networks as a system of queues, where the delay is determined by capacity and current usage. Gerla and Kleinrock (1977) derive the calculation for average packet delay based on this system of queues using the well-known Little's formula (Little (1961)). This calculation is used frequently – see Gavish and Neuman (1989) for a typical example. Also, the reader is referred to Nain and Ross (1992) for a special issue journal on queueing networks applied to transportation aspects of communication networks. Of recent importance, Boucherie and *van* Dijk (2000) present an equivalent queueing network description of a cellular mobile communications network.

Average packet delay is traditionally modelled in two main ways. Gavish and Neuman (1989) and Amiri *et al* (1999) have each unit of delay incurring a cost in the objective function. However, more frequently, a constraint is added that ensures average delay does not exceed a given maximum. For example, Gerla and Kleinrock (1977) and Dutta and Kim (1996) use this method.

Because delay is dependent on capacity and usage, Magnanti and Wong (1984) state that "communication systems are more likely to require non-linear models to represent congestion effects adequately". The literature tends to either approximate this non-linearity, or solve the problem heuristically, to find solutions to this complex model. However, in a recent paper, Le Blanc *et al* (1999) argue that because:

> *"In modern networks, heterogeneous traffic (interactive video, imaging, voice, and data) have very different delay and jitter tolerances, admission control rules, discard policies, etc,...simple queueing formulae in the flow λ and capacity μ such as $\lambda / (\mu - \lambda)$ are less appropriate, and accurate analytical representation of actual queueing is extremely complex. Instead, models that indirectly consider queueing by limiting flows to node effective capacities are important."*

To our knowledge, Le Blanc *et al* (1999) are the first to consider the underlying queueing networks and congestion indirectly. They do so by restricting flow to an 'effective capacity' – the level of flow where congestion is such that quality of service deteriorates beyond some pre-defined minimum level. In this thesis we also consider the underlying queueing networks indirectly, providing a comparison with Le Blanc *et al*'s method.

2.4. Internet Research

The global transfer of information using the Internet is a recent phenomenon. From its humble beginnings as ARPANET (Advanced Research Projects Agency Network), a network set up by the US Department of Defense in 1969 in conjunction with four UCLA campuses, the Internet has boomed, particularly in the past ten years[1], to the extent that it now seems irreplaceable. However, this success has caused problems, namely the burden of congestion, which at times can cripple the Internet. Thus research into ways of easing this congestion has become a significant focus of the literature. Section 2.4.1 provides an overview of this research. Significant literature also exists, both in peer-reviewed journals and the popular press, on how to use the

[1] Sampat (2000) reports that the number of Internet host computers, a reliable measure of Internet size, has grown from less than one million in 1991 to more than seventy million in 1999. Duffy (1998) reports that Internet traffic doubles every six months.

Internet to the firm's best advantage – Internet marketing and Internet commerce systems are common examples. We do not review this literature.

2.4.1. Congestion and Solutions

Internet congestion is caused by excessive flow being sent through routers and over links of limited capacity. As the utilisation of these network elements increases, congestion occurs, and throughput slows as a result. This problem has been predominantly caused by the aforementioned success of the Internet. Usage levels have surged, and capacity has struggled to keep up. Exacerbating this problem is that users, unhampered by 'all-you-can-byte' flat-rate pricing structures, tend to act greedily, as they "see only the limitless possibilities for information access and have no idea about the cost" (Shotsberger (1996)). Adding to this problem is that increasingly complex applications, which require more capacity, are being developed, and hence demanded. Numerous solutions to Internet congestion have been suggested, and can be categorised regarding: network technological improvements, pricing schemes, and modelling network expansion.

Network Technological Improvements

The most investigated solution to easing the strain on the Internet's resources is to improve technological aspects of the network. Because of its size, and particularly the speed of its growth, Internet routing protocols contain inefficiencies. Lawson (1996) discusses how Internet routers are slowed by having to form enormous routing tables and conduct complex calculations that now exist because of the Internet's size. Higgins and Woods (1998) discuss backhauling, where a significant proportion of all global traffic is slowed by being routed back to the "US Internet cloud". Duffy (1998) asserts that new and improved routers are critical.

Savage *et al* (1999) discuss poor routing systems of current protocols which can send packets along sub-optimal routes, partly because often not all route information is available for decision-making. They offer ways of fine-tuning these systems. Kelly (2000) also suggest modified routing systems which are more intelligent when

forwarding packets. Other routing technology (Lind (1998)) is designed to bypass major Internet congestion areas. Slightly differently, Chapman and Kung (1998) and Dryden (1996) look at updated Internet protocols. Lee *et al* (2002) discuss how Internet traffic can benefit from the congestion control mechanisms of Internet protocols (TCP / IP).

Carpenter and Kandlur (1999) suggest that one of the major current problems with Internet delivery at the moment is that every packet receives the same quality of service. Quality of service in future networks should be dependent on the application being requested (for example, immediate audio / video transfer versus overnight file transfer) and on how much the customer pays.

Two other potential solutions to network congestion, not possible in telecommunications networks, are load balancing and caching. Load balancing, as discussed by Adhikari (1998), improves Internet performance by attempting to evenly distribute a system's entire load over all the potential servers in the system. Caching stores recently used information on the user's computer (in a limited-memory cache) in the hope of retrieving that information from there should it be required again, rather than once more having to incur network flows to do so (adding to congestion). Baentsch *et al* (1997) discuss how caching works, both at an overview and technical level. Mookerjee and Tan (2002) is an example of research that analyses different methods of cache management (which downloaded files to remove from the cache when newly cached files arrive).

Pricing Schemes

The advent of flat-rate, inexpensive, Internet pricing schemes (such as in Lewis (1996)) has added to the congesting of the Internet. Under these schemes, there are no incentives for an individual to restrict their Internet usage. From a global perspective this is very costly. Huberman and Lukose (1997) argue that congestion problems would disappear if individuals were charged in proportion to their bandwidth consumption.

Network Modelling

Because the Internet is a disorganised collection of interconnected self-organising networks, modelling of the Internet to improve performance is complex. Calvert *et al* (1997) review the basic topological structure of the Internet, as well as presenting a modelling method designed to produce graphs that reflect the locality and hierarchy present in the Internet. These network graphs are important for, amongst other things, the testing of routing algorithms. Cleveland and Sun (2000) look at statistical models which attempt to generate traffic that mimics the behaviour of Internet links closely. This is a difficult process given the 'bursty' nature of Internet traffic. Cowie *et al* (1999) focus on simulating models of the Internet. Because of the Internet's size, these models are significantly downscaled, but still need to be large enough to allow the drastic fluctuations that seem to regularly appear in the Internet to occur. Crowcroft (2000) models quality of service for Internet applications, particularly outlining the differences caused by the Internet's 'messy' design, compared with the ground-up controlled design of telecommunications networks.

2.5. Expansion of Processing Resources

As discussed in Chapter 1, while the need to provide processing resources has increased, there is little research into the optimal expansion of processing resources. However, whilst in its formative stages, some research has looked at the best use of available processing resources. As far as we are aware this is a complete review of this area.

In the setting of one of the world's largest international data networks[2], Kreidi and Sanso (1994) look at the problem of finding the optimal computer processing size, and the optimal strategy for transferring traffic between those computers, given the geographical location of the system and the characteristics of data input (such as demand). Assuming the load transferred between computers over the high-speed

[2] SITA, which is a leading provider of global information and telecommunication solutions to the air transport industry.

telecommunication network experiences negligible delay, Kreidi and Sanso develop a mixed-integer program to represent this problem. Non-linearity is present in the constraints, to ensure response times do not exceed an allowed level. A solution approach using decomposition is proposed. Importantly, whole jobs are processed on a single machine; "processing of a message is never split between two sites".

With the increased flexibility of a distributed processing environment, jobs can be split and distributed throughout the network. Tomasgard *et al* (1997) and Tomasgard (1998) are the first that we are aware of to look at processing allocation decisions in this environment. They also briefly investigate the strategic decision that sets that processing capacity.

Tomasgard *et al* (1997) describe new operational and strategic optimisation models for distributed networks. After outlining the distributed processing framework, they focus on modelling variations of the operational problem, that being how to best utilise network resources to meet demand. Following this they formulate stochastic integer programs that look at the strategic problem of setting the total amount of network processing capacity. Using the solution to this strategic problem as the lower bound for total network processing capacity, a node location model is formulated to decide where processing capacity should be installed in the network to minimise the combined investment and operational costs. Rejections are allowed at the second stage if the realised demand exceeds capacity. Rejecting subservices is viewed as paying for another service provider to provide that subservice (the rejection cost). This research is conducted assuming processing capacity is the limiting resource, and that "demand for processing capacity can be served at any node independent of the location of demand, without reducing quality of service". The underlying assumption, following Kreidi and Sanso (1994), is that the transportation networks have enough capacity to ensure time delay for transportation of information is negligible.

Like the previous paper, Tomasgard (1998) also focuses on distributed processing aspects and distributed services' use of computing resources at network nodes. This includes discussing how the change in focus in service provision towards the processing of information will influence optimisation models. Uncertainty in demand is captured in the operational service provision models, with a two-stage stochastic

integer program being formulated. Models were developed under the same transportation assumptions as for Tomasgard *et al* (1997). Based on these models, optimisation algorithms and approximation heuristics were developed to solve the problem where demand was (1) deterministic, and (2) described discretely in terms of scenarios. The deterministic problem was investigated, even though demand was identified as being uncertain, to learn about the problem's complexity and combinatorial aspects. When rejections and stochasticity were considered the problem was shown to be strongly NP-hard. Because of this approximation heuristics were found, as with this class of problem there is little hope of finding efficient exact methods. However, for comparison purposes, optimal solutions were found for small network problems using dual decomposition (scenario decomposition) and Benders' decomposition. Tomasgard *et al* (1998) is a published paper from this thesis, predominantly describing the initial service provision problem.

2.6. Summary

Capacity expansion is a well-studied problem. From the early work by Manne (1961) much research has been conducted modelling aspects of, and finding solutions for, the expansion problem. This chapter has reviewed this literature, particularly the aspects relevant to this thesis: stochasticity, modelling of demand, different technologies, congestion costs, and service delay.

As setting capacity is a general problem, research in the literature has been conducted in many industries, from transportation and electricity networks, to the communication networks studied in this thesis. From the communication networks' point of view, there exists a significant amount of research. Because the routing of flow has traditionally been the cause of network bottlenecks, most of this research concentrates on the network routing expansion problem. Recently, however, some research has begun to investigate the processing side of the network. This problem has grown in importance because new multimedia services require more computing resources, making these a potential bottleneck. Also, technological advances in routing technology have alleviated that bottleneck slightly; however, we recognise the problem of routing congestion will always be present.

Kreidi and Sanso (1994) look at the processing side of the network, where whole jobs are processed at a single machine. Given the development of distributed processing networks, where jobs can be more efficiently met by being split and distributed over the network, Tomasgard (1998) was the first to develop optimisation models considering processing-based services in this environment. This research did not consider transportation and assumed that quality of service was independent of where and how demand was met.

Our review of the literature has highlighted the need for further research into the processing capacity expansion decision, while considering transportation and quality of service issues. Having identified this, we now proceed to develop models for this problem.

3. PROCESSING DECISIONS

The operational processing decisions meet demand in the environment set by the strategic capacity decisions. As such, in order to model the capacity investment decisions, it is crucial to consider their impact on the processing decisions.

3.1. Introduction

The processing decisions are frequent operational decisions that, given fixed levels of processing and routing resources, determine what and how demand will be met. Specifically, as outlined by Dye *et al* (1998), the processing decisions determine which processing nodes to run the subservices on. This decision must be made considering (an approximation of) how the demand received would be met once the subservices were installed. Using mathematical models can improve this decision-making. There are two important reasons for developing operational decision models:

- To help the strategic decision makers when determining what resources to provide. Sridharan (1995) and Karmarker and Kekre (1987) discuss the

importance of considering the operational environment when making capacity decisions. Because of their importance to the capacity decision, the processing decisions deserve to be investigated in detail.

- To provide the operational decision makers with solutions to the problem of which service requests to accept and how to meet them, given fixed resources. Whilst being an important aspect of the capacity decision, the processing decisions are also important in their own right; as once the capacities are set, these decisions must be made.

It is important to note that different operational models are used for the above two purposes. This chapter develops an operational model for a single processing decisions period in isolation. Whilst this model is used as the basis for the representation of the operational environment in the later capacity model, important subtle conceptual changes need to be made for this purpose. These changes are discussed in Chapter 5.

This chapter firstly investigates the problem situation, identifying in Sections 3.2 and 3.3 the relevant factors and relationships the decision maker should consider when making the processing decisions, before formulating a mathematical model for these decisions in Section 3.4. Because the aim of this chapter is to aid insight into the strategic capacity expansion decision, a number of complicating factors, such as stochasticity, which would need to be considered if this model were to be used in reality, are not considered at this stage.

3.2. Problem Situation Part I: Diagrammatic Overview

The operational decision is how to best use the network resources available to meet demand requests. Figure 3.1 presents an influence diagram for the system of interest (influence diagram notation used is as defined in Daellenbach (2001)). It shows all the factors and relationships that should be considered in this decision-making environment. This includes those factors that are outside the service provider's control in this problem situation (uncontrollable inputs), including those that the service provider controls at the strategic level, such as capacity. Also included are the

inputs resulting from decisions made in this environment (control inputs) and the system variables (the results given both sets of inputs).

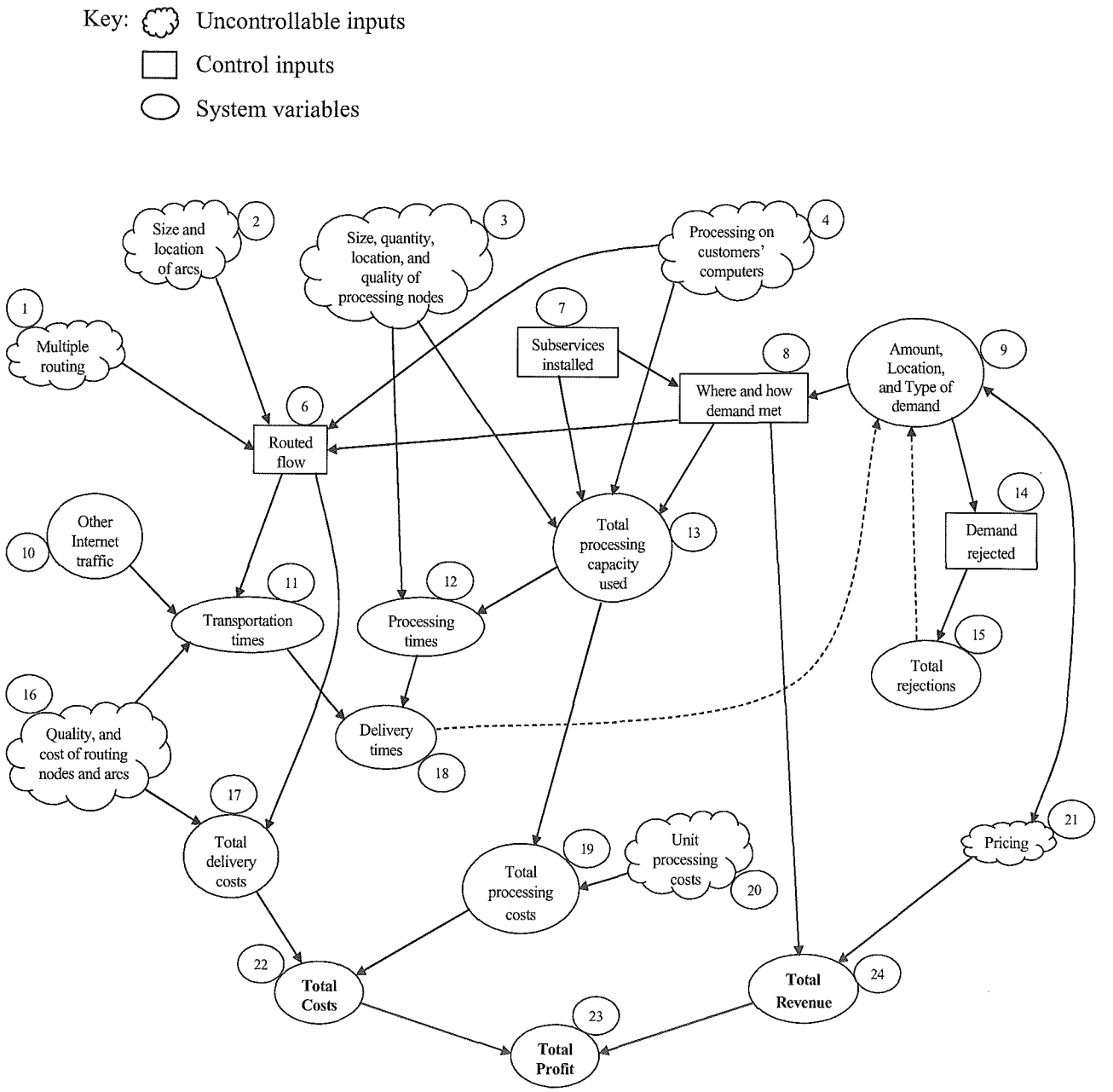


Figure 3.1. Influence diagram representing the processing decisions problem situation.

Given fixed resources (processing and routing), the service provider must decide what service requests to meet, and how to meet them. Specifically, this involves determining where to install subservices, what demand to meet on which node, and what routes to take to move the associated flows. Demand for a subservice can only be met at a node if the subservice is installed at that node. Meeting demand earns revenue, but also incurs costs, and due to a feedback loop, discussed later, can influence future demand.

3.3. Problem Situation Part II: Detailed Discussion

This section provides a detailed discussion of all the factors and relationships in the system identified in the influence diagram (Figure 3.1). In the following discussion, numbers appearing in brackets refer to the part of the influence diagram currently being discussed. The discussion will begin with the processing decisions profit, and then work through the costs and revenues this consists of.

3.3.1. Processing Decisions Profits

In order to determine where to install subservices (7) and how best to meet demand (6, 8) the quality of the solutions are measured by the period's profit (23). Revenue (24) is obtained from meeting demand. The service provider may have a contract which includes pay-per-usage costs, and hence total costs (22) consist of total per-unit processing (19) and routing (17) costs, as well as implicit costs penalising poor quality of service (see Section 3.3.4).

3.3.2. Resources

The processing decisions are made under fixed resources, which are set by the service provider at the strategic level. These include: the routing network (arcs and routing nodes and their characteristics: capacities, costs, and quality) (2, 16), the technology available, the services provided, and the processing resources available (and their characteristics) (3, 20). The amount of demand that can be met is limited by the processing and routing resources provided.

3.3.3. Demand and Price

A period's demand is influenced by many factors, such as price and the quality of service provided by the service provider in previous periods. Over the longer time horizon of the capacity investment decisions we look in more detail at how these factors influence demand. However, in this short time-frame the impact these factors have on the period's demand has already occurred and is hence already included in our demand estimate. As mentioned previously, both price (21) and demand (9) are included deterministically in the processing decisions model, where realistically they are likely to be stochastic. A period's demand is assumed to occur at the beginning of that processing decisions period.

3.3.4. Operational Costs

Much of the service provider's investment in network infrastructure are sunk costs, hedged against profits from meeting demand both now and in the future[3]. Hence, to make this investment worthwhile it is important that the processing decisions ensure adequate levels of demand in the future. Because of the large investment costs, it is likely that minimising costs in the current period is inadvisable as this may jeopardise future demand.

The way demand is met (8) determines the quality of service received by customers. If the quality of service is poor, customers become dissatisfied, and future demand will likely decrease (Stuart and Tax (1996)). Hence, an inherent feedback loop is present, where the level of current demand, and how it is met, influences the level of future periods' demand. However, as we are only considering one processing decisions period at this stage, the dashed feedback loops in the influence diagram are broken in this model. Hence, to ensure future demand is protected in the single-period model, poor quality of service must be penalised. This is done by placing a

---

[3] The service provider may also, however, have pay-per-unit arrangements for processing (20) and routing (16) capacity used.

cost on rejections (15) and delivery times (18) in the model. This idea is well summarised by Dewan and Mendelson (1990):

> *"Costs associated with service delays are usually intangible, and consequently they are frequently ignored or their impact underestimated. In many environments, however, timeliness is critical and delays entail a significant cost incurred by the user organisation"*

Complicating matters further, there is an inherent trade-off between rejections and delivery times, our measures of quality of service. This is because if all demand is met, no customers will be dissatisfied due to rejections; but customers could be dissatisfied because of the longer delivery times associated with meeting all demand. If demand is rejected the rejected customers are dissatisfied, but the remaining customers have increased satisfaction because their demand is now being met more quickly. The final trade-off chosen depends on the respective costs placed on rejections and delivery times.

Rejection Costs

The rejection costs are included in the model as a unit cost for each demand request rejected (14). The unit cost of rejecting a customer should be based on the expected loss in future profits from rejecting that customer and making them dissatisfied. This will depend on a customer's discounted future profit potential, the probability they will not return in the future, and how they may influence other customers. Therefore, different customers and different services will have different rejection costs associated with them.

Delivery Time Costs

The delivery time costs are included in the model as a constant cost for each unit of delivery time (18). This cost would again depend on the expected loss in future profits, this time associated with slow delivery times. Section 3.3.5 looks at how the delivery time for a service is determined.

3.3.5. Delivery Times

The delivery time (18) for a customer's service request depends on the time it takes to process the request (12) and the time it takes to route the resulting flows (11). Subservices are routed and processed concurrently, with on-going coordination. For example, a videoconferencing service consists of video and audio components. In a distributed processing environment these components are processed separately. However, in order for picture and sound to stay in-sync, the separate components must coordinate throughout their transportation. At various points in time the different components may need to wait to synchronise with each other. This means the delivery time for the service is somewhere between the time of the longest subservice and the sum of times for all subservices.

To reduce complexity, this is approximated in the model. We view a service's processing and transportation requirements as being independent until they are combined to produce the overall service. Hence, possible approximations include the sum or the average of the delivery times for all subservices. Using the average delivery time is the most accurate approximation when subservices take similar times to be processed and routed, whereas using the sum of all delivery times is most appropriate when delivery times for individual subservices are diverse. We use the sum of all subservices' delivery times in the model.

Processing Times

The more processing capacity being used at a node (13) the more processing congestion, and hence the longer the processing time for each request. The resulting total processing time function is a continuous increasing convex function, for example, Figure 3.2. Dewan and Mendelson (1990) show that aggregate service time functions exhibit this behaviour for a large class of service measures. This is because an extra unit to service not only adds the time to meet that unit, but also adds to overall congestion and so slows all service times (the marginal service times are increasing).

As discussed in Chapter 2, the majority of the existing literature (for example, Gerla and Kleinrock (1977)) models congestion using queueing theory[4]. Le Blanc *et al* (1999) state that accurate analytical representation of queueing is extremely complex in modern networks, and hence models that indirectly consider queueing are important. Such a model is used in this research, as we consider congestion by modelling the total processing time function (Figure 3.2).

Modelling Processing Times

Because subservices are installed before demand is requested, the customers only see delay due to the meeting of demand. Hence, for the processing decisions model we want to know the total processing time devoted to meeting demand[5], D_T.

Following is the explanation of how D_T is modelled for each network processing node.

X – processing capacity used meeting demand (model variable),

Z – processing capacity overhead for installing subservices (model variable),

D_T – total processing time devoted for meeting demand,

T(Y) – total processing time for processing Y units of capacity.

---

[4] The existing literature tends to model the transportation side of the network. However, by the same rationale (queues form when usage exceeds capacity) similar formulae can be used to model the processing times.

[5] Because the time spent processing demand is influenced by the amount of processing capacity used installing subservices (as this also adds to congestion), we should strictly represent this as $D_T(X:Z)$. For notational simplicity we use D_T.

We follow Le Blanc *et al* (1999) by modelling processing time behaviour as an M/M/1 queue. Hence, for Figure 3.2 the T function is of the following specific form (C is node capacity)[6].

$$T(Y) = \frac{Y}{C-Y}$$

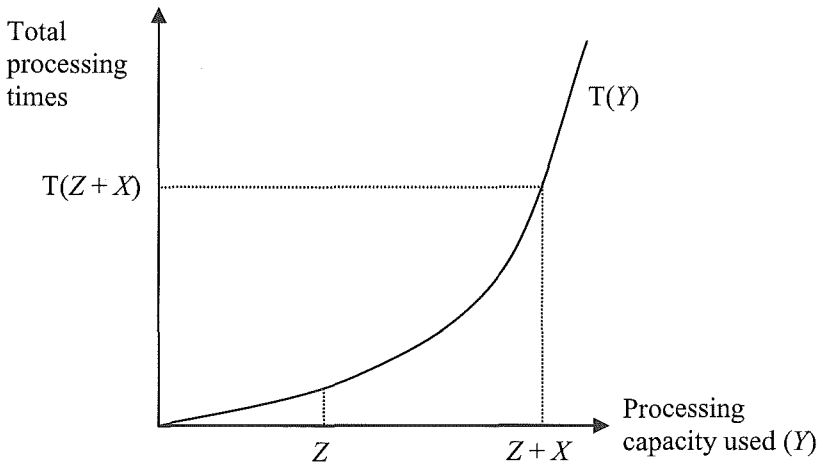


Figure 3.2. Total processing time function.

The total processing time at the node is $T(Z + X)$, and as each unit is processed at the same speed, each unit requires $T(Z + X) / (Z + X)$ processing time. Meeting demand uses X units, so the aggregate service time is:

$$D_T = T(Z+X) \frac{X}{Z+X}. \qquad (1)$$

The subservice overhead variables (Z) do not directly influence a customer's delivery time. However, by adding to nodal congestion, these Z variables cause all customer requests to be processed slower (by pushing the $Z + X$ point further up the curve).

---

[6] While we assume an M/M/1 queue in the model, all that is required is for T to be a continuous increasing convex function.

Obtaining the total processing times using this method is non-linear (X and Z are model variables). We use the linear approximation:

$$D_T \approx \mathrm{T}(Z+X) - \mathrm{T}(Z). \tag{2}$$

This approximation gives an over-approximation of the total processing times, as shown below.

$\frac{\mathrm{T}(Z+X)}{Z+X} y$ describes the line segment from T(0) to T(Z + X), for $y \in [0, Z+X]$, since T(0) = 0.

Since function T is convex:

$$\frac{\mathrm{T}(Z+X)}{Z+X} Z = \mathrm{T}(Z). \tag{3}$$

Also,

$$\frac{\mathrm{T}(Z+X)}{Z+X} X = \frac{\mathrm{T}(Z+X)}{Z+X}(Z+X) - \frac{\mathrm{T}(Z+X)}{Z+X} Z. \tag{4}$$

Hence, from (3) and (4), it follows that:

$$\frac{\mathrm{T}(Z+X)}{Z+X} X = \mathrm{T}(Z+X) - \mathrm{T}(Z). \tag{5}$$

Over-estimating processing times is better than under-estimating processing times, because profits are now a lower bound. Model users would prefer a more favourable outcome than the model predicted than a less favourable one.

Transportation Times

The transportation time for meeting demand consists of the time routing over arcs and through routing nodes. Congestion can occur at routing nodes. Like the processing times, we model the total transportation time through routing nodes as a continuous increasing convex function, also of the M/M/1 queue form: flow / (capacity – flow). In this case the delay time must consider general Internet traffic (10) as well.

We assume arcs transport information at the same speed, regardless of the volume on the arc, until their capacity is reached. This is modelled as a constant transportation time for each unit of flow routed over an arc.

In this chapter it is assumed that the processing and routing times for a given amount of processing / flow are known with certainty. However, even if the service provider knows the processing (and routing) time functions with certainty, the actual processing (and routing) times are likely to be stochastic. This is because over the time it takes to meet a service, the processing (and routing) required will be 'bursty', using different amounts of processing (and routing) at different times throughout the process. This stochasticity is considered in Chapter 8.

3.3.6. Other Factors

There are a number of other factors that can affect how a service provider meets demand. For various reasons, discussed below, these are not considered in this processing decisions model.

Processing on Customers' Computers

It is possible, given the current state of technology, that customers actually perform some of their own processing (4). For this to occur the service provider must decide at the strategic level to develop services to allow this. This extension is considered in Chapter 10.

Caching

As discussed in Chapter 2, caching is a method for dealing with poor Internet quality of service, and as such could be applicable in these models. However, caching improves Internet performance by keeping frequent and recently retrieved information close to the customers, reducing delivery times. Caches are hence not appropriate in this environment because service requests are unique, and hence recently processed information is of no future use.

Multiple Routing

A service provider routes flow destined for another processing node over the route between them with the lowest delivery time. In practice, delivery times are not known for sure, and the route with the lowest expected delivery time would be chosen. To hedge against possible long delivery times for this route the service provider could send the flow over multiple routes (1). This would have the trade-off of increasing network congestion. We do not consider this aspect.

3.4. Mathematical Model (PDS model)

The previous sections defined the problem situation for the single period processing decisions, and discussed how the various factors in the problem situation were to be included in the mathematical model. This section formulates the one-stage deterministic mathematical model (PDS model) for this problem situation.

The decisions are: what subservices to install, what demand to meet and reject, where to meet demand, and what routes to take doing so. An exhaustive list of the indices and sets for the problem are presented in Table 3.1, the parameters (lower case) in Table 3.2, and the decision variables (upper case) in Table 3.3. The constraints and objective function for the problem are then presented and discussed, before the entire formulation is given in Figure 3.4.

Indices

a = arc.

dr = location where demand is received.

k = subservice.

n = potential processing location.

r = route.

rn = routing node.

s = service.

Sets

J_a = routes using arc a.

K_{rn} = routes through routing node rn.

L_k = services that use subservice k.

$R_{dr,n}$ = routes between location dr and processing node n.

Table 3.1. Indices and sets for the processing decisions model.

Parameters

b_n = the pay-per-usage unit cost of using processing location n.

c = the penalty 'cost' of one unit of delivery time.

$d_{s,dr}$ = the amount of demand for service s received at location dr.

f_n = the total processing capacity available at processing node n.

$h_{k,s}$ = the processing units of subservice k used by a request for service s.

j_k = the number of units of flow required to route one unit of subservice k.

$l_{s,dr}$ = the penalty 'cost' of rejecting service s received at location dr.

m = a large number.

$p_{s,dr}$ = the price obtained for meeting a request for service s received from location dr.

q_a = the capacity of arc a.

t_a = the time it takes to send a unit of flow over arc a.

u_k = the amount of processing capacity it takes to install subservice k.

v_a = the pay-per-usage unit cost of routing a unit of flow over arc a.

w_{rn} = the pay-per-usage unit cost of routing a unit of flow through routing node *rn*.

Table 3.2. Parameters for the processing decisions model.

Decision variables

P = sum total of times taken to process all subservices.

$R_{s,dr}$ = units of demand rejected for service *s* received at demand location *dr*.

T = sum total of times taken transporting all the flow.

$W_{dr,n,r}$ = flow between location *dr* and processing node *n* routed over route *r* between those two locations.

$X_{k,dr,n}$ = processing units used by subservice *k* for meeting demand originating from location *dr* which is processed using computer *c* at location *n*.

$Y_{s,dr}$ = units of demand for service *s* received at location *dr* that are actually met.

$Z_{k,n}$ = 1, if subservice *k* is installed at processing location *n*, 0, otherwise.

Table 3.3. Decision variables for the processing decisions model.

Constraints

Demand Constraint:

$$Y_{s,dr} + R_{s,dr} = d_{s,dr} \qquad \forall\, s,dr, \qquad (1)$$

Constraint (1) ensures that demand for service *s* received at location *dr* is either met or rejected. Unlike Tomasgard (1998), we reject full services, not subservices.

Processing Translation Constraint:

$$\sum_{s \in L_k} h_{k,s}\, Y_{s,dr} = \sum_{n} X_{k,dr,n} \qquad \forall\, k,dr, \qquad (2)$$

The left hand side of Constraint (2) is the total processing requirement for subservice k, coming from all services that use that subservice, and the right hand side is the total processing for subservice k performed in the network.

Subservice Installation Constraint:

$$X_{k,dr,n} \leq Z_{k,n}\, m \qquad \forall\, k,dr,n, \qquad (3)$$

Constraint (3) ensures that demand for subservice k can only be met at processing node n if the necessary subservice is installed there. Note, for solution purposes, it is important to ensure m is as small as possible, without further limiting $X_{k,dr,n}$, such as being based on the amount of demand.

Processing Node Capacity Constraint:

$$\sum_{k}\sum_{dr} X_{k,dr,n} + \sum_{k} Z_{k,n}\, u_k \leq f_n \qquad \forall\, n, \qquad (4)$$

Constraint (4) ensures that the processing capacity used at node n does not exceed the capacity available at that node. Capacity is used installing subservices and meeting demand.

Flow Requirement Constraint:

$$\sum_{k} j_k\, X_{k,dr,n} = \sum_{r \in R_{dr,n}} W_{dr,n,r} \qquad \forall\, dr,n, \qquad (5)$$

This constraint ensures that the flow resulting from demand received at location dr and met at processing node n is routed between those two points. It is assumed that the flow generated by a subservice is independent of the service generating that subservice request.

Arc Capacity Constraint:

$$\sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \leq q_a \qquad \forall\, a, \qquad (6)$$

Constraint (6) ensures that the level of flow on an arc does not exceed the capacity of that arc. The left-hand side defines the level of flow over an arc as being the sum of the flow over routes containing that arc. Note that there are other ways of including the routing of flow in the model. This technique was used for notational convenience.

Processing Times Calculation:

$$P = \sum_{n} F_n(f_n, \sum_{k} \sum_{dr} X_{k,dr,n}, \sum_{k} Z_{k,n}\, u_k) \qquad (7)$$

P is the total aggregate processing time over all nodes. The total processing time at node n depends on the amount of processing capacity available, the amount being used meeting demand, and the amount being used to make subservices available. Different nodes may have different processing rates and breakpoints (different F_n). Aggregate processing times are found by summing the processing times for the respective processing nodes.

Referring back to the notation used in Section 3.3.5, F_n is defined as (where X_n represents $\sum_{k} \sum_{dr} X_{k,dr,n}$, and Z_n represents $\sum_{k} Z_{k,n} u_k$):

$$F_n(f_n, X_n, Z_n) = \mathrm{T}(f_n, X_n + Z_n) - \mathrm{T}(f_n, Z_n)$$

Where, $$\mathrm{T}(f, y) = \frac{y}{f - y} \qquad \text{(for an M/M/1 queue)}$$

Hence, $$F_n(f_n, X_n, Z_n) = \frac{X_n + Z_n}{f_n - X_n - Z_n} - \frac{Z_n}{f_n - Z_n}$$

Like Hiller and Shapiro (1986), to reduce complexity we model the total processing time function (Figure 3.2, F_n) using a convex piece-wise linear approximation. The advantage of doing this is that the formulation keeps its mixed-integer nature, rather than becoming non-linear. However, binary variables are required to incorporate the subtraction of T(Z) (see Section 3.3.5). Appendix C details the exact code for these piece-wise linear functions, including the breakpoints and gradients used.

Figure 3.3 shows that non-convexity is created from the binary variables necessary for subservice installation, not those for the processing time function.

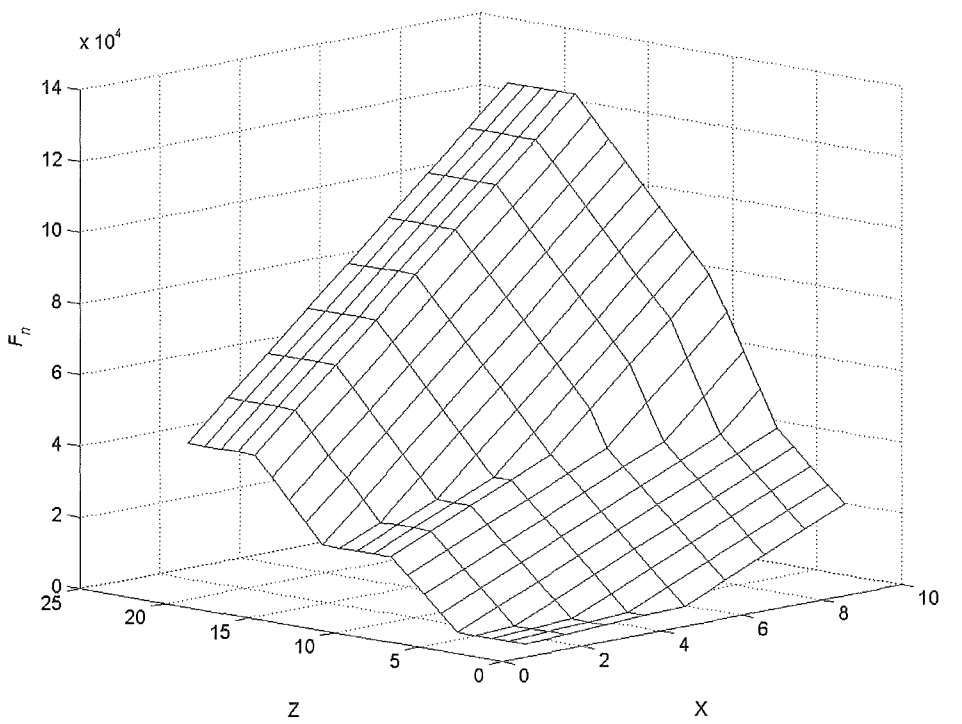


Figure 3.3. Non-convex plot of total nodal processing time, F_n, versus processing capacity used to meet demand (X) and install subservices (Z).

Transportation Times Calculation:

$$T = \sum_{rn} H_{rn}\left(\sum_{dr,n} \sum_{r \in R_{dr,n}; r \in K_{rn}} W_{dr,n,r}\right) + \sum_{a} t_a \sum_{dr,n} \sum_{r \in R_{dr,n}; r \in J_a} W_{dr,n,r} \qquad (8)$$

This constraint calculates the total transportation times. This consists of the time spent routing through nodes (first term) and over arcs (second term), which is dependent on the amount of flow sent over routes containing those components. The H_{rn} function is as described by Le Blanc *et al* (1999) for an M/M/1 queue. This is also modelled using a convex piece-wise linear approximation. However, binary

variables are not required to model this function. For simplicity, the explicit piece-wise linear function is not shown here (see the code in Appendix C).

Objective Function

$$\begin{aligned} \textit{Max} \quad & \sum_{s}\sum_{dr} Y_{s,dr}\, p_{s,dr} - \sum_{n} b_n \Big(\sum_{k}\sum_{dr} X_{k,dr,n} + \sum_{k} u_k Z_{k,n}\Big) \\ & - \Big(\sum_{a} v_a \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} + \sum_{rn} w_{rn} \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in K_{rn}} W_{dr,n,r}\Big) \\ & - \sum_{s}\sum_{dr} R_{s,dr} l_{s,dr} - (P+T)\, c \qquad (0) \end{aligned}$$

Objective (0) represents the processing decisions period profits. The first term is the revenue obtained from meeting demand. The second and third terms are the pay-per-usage costs of processing and routing respectively. The final two terms are the costs penalising rejections and delivery times.

Formulation

$$\begin{aligned} \textit{Max} \quad & \sum_{s}\sum_{dr} Y_{s,dr}\, p_{s,dr} - \sum_{n} b_n \Big(\sum_{k}\sum_{dr} X_{k,dr,n} + \sum_{k} u_k Z_{k,n}\Big) \\ & - \Big(\sum_{a} v_a \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} + \sum_{rn} w_{rn} \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in K_{rn}} W_{dr,n,r}\Big) \\ & - \sum_{s}\sum_{dr} R_{s,dr} l_{s,dr} - (P+T)\, c \qquad (0) \end{aligned}$$

s.t. Constraints

$$Y_{s,dr} + R_{s,dr} = d_{s,dr} \qquad \forall\, s,dr, \qquad (1)$$

$$\sum_{s \in L_k} h_{k,s}\, Y_{s,dr} = \sum_{n} X_{k,dr,n} \qquad \forall\, k,dr, \qquad (2)$$

$$X_{k,dr,n} \le Z_{k,n}\, m \qquad \forall\, k,dr,n, \qquad (3)$$

$$\sum_{k}\sum_{dr} X_{k,dr,n} + \sum_{k} Z_{k,n}\, u_k \le f_n \qquad \forall\, n, \qquad (4)$$

$$\sum_{k} j_k \, X_{k,dr,n} = \sum_{r \in R_{dr,n}} W_{dr,n,r} \qquad \forall\, dr,n, \qquad (5)$$

$$\sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \leq q_a \qquad \forall\, a, \qquad (6)$$

$$P = \sum_{n} F_n(f_n, \sum_{k} \sum_{dr} X_{k,dr,n}, \sum_{k} Z_{k,n} \, u_k) \qquad (7)$$

$$T = \sum_{rn} H_{rn}(\sum_{dr,n} \sum_{r \in R_{dr,n}: r \in K_{rn}} W_{dr,n,r}) + \sum_{a} t_a \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \qquad (8)$$

$$Z_{k,n} \in \{0,1\},$$

$$R_{s,dr},\ X_{k,dr,n},\ Y_{s,dr} \geq 0.$$

Figure 3.4. Formulation for the single-period processing decisions model (PDS).

3.5. Summary

The processing decisions determine how to make best use of the available computational resources when meeting demand for services. This chapter described the system of interest for these processing decisions by presenting an influence diagram and then providing a detailed explanation of the factors and relationships. The two most important points to come from this are as follows:

- The need to penalise poor quality of service by including a 'cost' for rejections and delivery times. This was necessary because poor quality of service leads to reduced future demand, affecting the return the service provider can make on their capacity investments. This modelling approach simplifies the complex dynamics that exist here. These are investigated and modelled in Chapter 6.
- Congestion means that the processing and transportation time functions are both convex. They are modelled using piece-wise linear functions, rather than analytical queueing formulae.

Following the explanation a deterministic mathematical model was developed (PDS model) to aid decision-making in this area. This model determines what demand to meet and how to meet it, given fixed processing and routing resources. Processing and arc capacity are the constraining factors. The remaining model parts (other than the necessary subservice installation constraint) determine costs and revenues. This model was formulated for a single processing decisions period, and subtle changes are required for use in the processing capacity expansion model. The necessary changes are outlined in Chapter 5. Before that, Chapter 4 analyses the most important aspects of the processing decisions model, with respect to producing appropriate solutions.

4. Analysis of the Processing Decisions Model

Investigating the processing decisions model, and its complexity, can provide insight as to what operational model detail is required to ensure good quality solutions from the capacity expansion model.

4.1. Introduction

The processing decisions were modelled in the previous chapter using a single-period deterministic integer model (PDS). In this chapter we assess what operational model complexity will likely be important for the capacity model to obtain appropriate capacity solutions. This is done by investigating model solution types. This chapter also provides some insight as to why less operational model complexity will likely reduce the quality of capacity solutions.

This chapter proceeds as follows. Section 4.2 defines categories of solution types. Using this information, Section 4.3 determines whether approximations to the PDS

model reduce the types of solution the model produces, and whether this is important. Finally, Section 4.4 discusses why alternative objective functions to that modelled in Chapter 3 are inappropriate.

4.2. Model Solutions

In order to compare approximations of the PDS model, and assess their appropriateness, it is important to investigate what different types of optimal solutions the PDS model produces. We want to know the solutions that occur without interference from capacity restrictions. This will show, for a given data set, what is the best network configuration had capacity been free. In the capacity model there will be pressure to move towards such a configuration, which will be traded-off against the costs of providing that configuration. Hence, the analysis in this chapter assumes infinite arc and processing capacity. The influence of restrictive capacities is discussed later in the chapter.

4.2.1. Solution Types

While investigating the PDS model, it was found that different model instances produced optimal solutions that could be grouped into one of three solution categories. The solution types (presented in an example in Figure 4.2) are:

- 'Fully centralised processing' solution, where all demand received from all locations is met at one processing node in the network.
- 'Centralised processing by node' solution, where all demand received from a location is met at one processing node in the network. Unlike the previous solution type, multiple processing nodes can be used meeting the demand received from different locations. The first type of solution is a special case of this solution type.

- Finally, 'distributed processing' solutions encompass all solutions. Unlike the previous solution types this includes when the meeting of a location's demand is shared, or distributed, between different processing nodes in the network. Multiple processing nodes can be used to meet the demand that is received from a particular location. The first two solution types are special cases of this solution type.

The solution space is shown diagrammatically in Figure 4.1. The solution types exhaust the solution space because 'distributed processing' is an all-encompassing solution type. In the following discussion, *strictly* 'distributed processing' solutions refer to the subset of distributed processing solutions that do not centralise their processing.

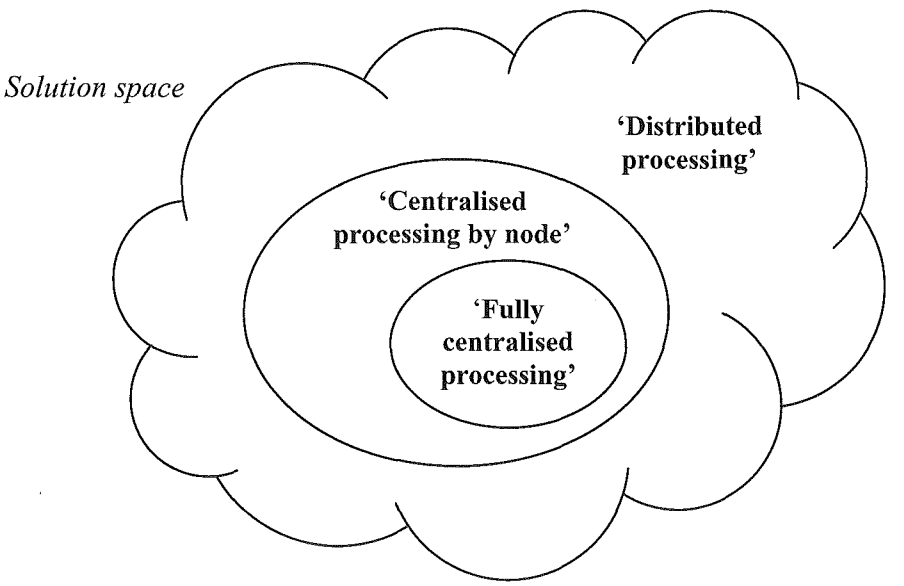


Figure 4.1. Diagram of solution space.

'Fully centralised processing' solution				
		Demand processed		
		1	2	3
Demand received	1	100%	0	0
	2	100%	0	0
	3	100%	0	0

'Centralised processing by node' solution				
		Demand processed		
		1	2	3
Demand received	1	100%	0	0
	2	100%	0	0
	3	0	0	100%

'Distributed processing' solution				
		Demand processed		
		1	2	3
Demand received	1	40%	40%	20%
	2	30%	30%	40%
	3	30%	20%	50%

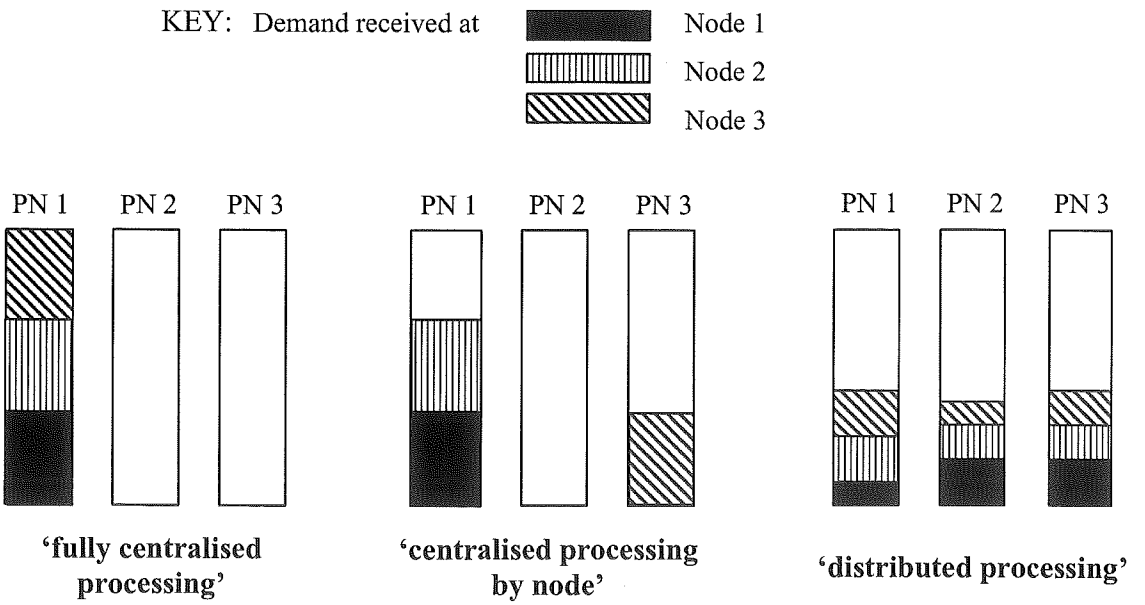


Figure 4.2. Example showing the three solution types.

4.2.2. How Solutions Types Occur

Centralised processing solutions occur when, for every unit of demand received at a location, the same processing node is the most profitable to meet demand at. This occurs when the marginal costs of meeting demand are relatively constant as the amount of demand met increases – otherwise it would soon become more profitable to use another processing node. Relatively constant marginal costs occur when meeting all demand produces little increase in congestion, and hence the delivery time costs are relatively constant (as all the other costs are constant per unit). 'Fully centralised processing' solutions occur when it is the same node that is the cheapest for all locations, whereas 'centralised processing by node' solutions occur when for different demand locations different processing nodes are cheaper (such as if transportation costs are high between particular demand and processing locations). A special 'centralised processing by node' solution is where all demand is met at the node where it was received; this leads to no transportation occurring between locations.

Strictly 'distributed processing' solutions occur when the cheapest node for meeting a location's demand changes as more demand is met. This occurs when there are increasing marginal costs, such as caused by congestion. Hence, it is the convexity of the delivery time functions that leads to distribution of processing. A special 'distributed processing' solution, occurring in times of extreme congestion, is where each subservice is only installed once in the network, and all demand for that subservice is met at that one node. Different nodes could be used to meet different subservices.

4.2.3. Mathematical Example

The previous section outlined, conceptually, how the different solution types may occur. This section builds on the previous one, using a simple mathematical example, where the parameter values for which each type of solution occurs are identified, to further add to the understanding of how the respective solution types occur.

The network for this example, and many of the parameter values, are shown in Figure 4.3. Only one service (composed of a single subservice) is available, which

can be demanded from either node. One unit of subservice requires one unit of processing capacity, and uses one unit of arc capacity. Installing the subservice does not use any processing capacity ($u_k = 0$). For the piece-wise total processing time function, the slope (e_j) represents the time per unit to process at a node. In addition, price is $p_{s,dr} = 30$, the rejection cost is $l_{s,dr} = 10$, and the delivery time penalty cost is one ($c = 1$). The costs associated with routing through nodes and congestion at routing nodes are zero.

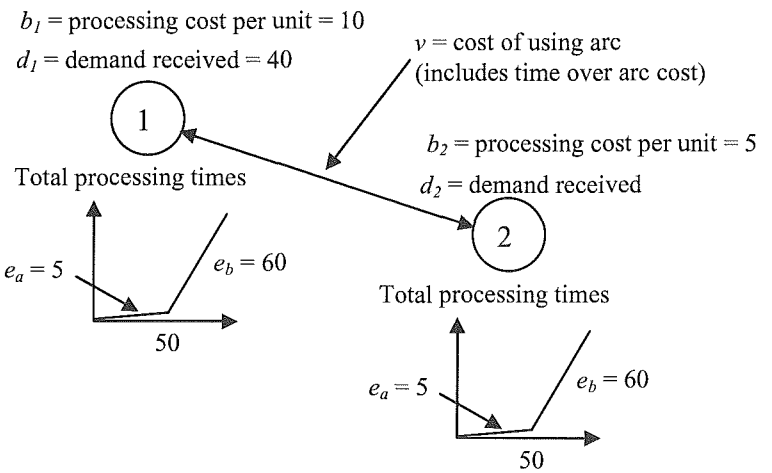


Figure 4.3. Simplified network for demonstrating PDS model solutions.

The resulting formulation of the PDS model for this simple example is shown in Figure 4.4. This formulation has no capacity constraints because we are assuming unlimited capacities at this stage of the analysis. The model solutions, given the respective values of v and d_2, are shown in Table 4.1.

Let:

$X_{dr,n,j}$ = demand received at node dr that is met at node n where the processing time is at rate e_j. The j index is a way of making the formulation a linear program. This modelling detail is not shown in Chapter 3's model.

Examples of the objective function co-efficient for this variable include: profit for $x_{11a} = p_{11} - b_1 - c\ e_a = 15$, and profit for $x_{21a} = p_{12} - b_1 - c\ e_a - v = 15 - v$.

R_{dr} = units of demand received at node dr that are rejected.

Model:

$$\textit{Maximise}\ 15\,X_{11a} + \text{-}40\,X_{11b} + (20 - v)\,X_{12a} + (\text{-}35 - v)\,X_{12b} + (15 - v)\,X_{21a}$$
$$+ (\text{-}40 - v)\,X_{21b} + 20\,X_{22a} + \text{-}35\,X_{22b} - 10\,R_1 - 10\,R_2$$

(maximise profit)

s.t.

$$X_{11a} + X_{11b} + X_{12a} + X_{12b} + R_1 = 40$$
$$X_{21a} + X_{21b} + X_{22a} + X_{22b} + R_2 = d_2$$

(meet nodal demands)

$$X_{1na} + X_{2na} \leq 50 \qquad n = 1,2$$

(capacity available at $j = a$)

Figure 4.4. Formulation for demonstrating PDS model solutions.

	$v = 5$		
	$d_2 \leq 10$	$10 < d_2 < 50$	$d_2 > 50$
Solution (non-zero variables)	$X_{12a}^* = 40$, $X_{22a}^* = d_2$.	$X_{11a}^* = d_2 - 10$, $X_{12a}^* = 50 - d_2$, $X_{22a}^* = d_2$.	$X_{11a}^* = 40$, $X_{21a}^* = \min\{10, d_2 - 50\}$, $X_{22a}^* = 50$, $R_2^* = \max\{0, d_2 - 60\}$.
Solution type	'fully centralised processing'	'distributed processing'	
	$5 = v = 30$		$v = 30$
	$d_2 \leq 50$	$d_2 = 50$	
Solution (non-zero variables)	$X_{11a}^* = 40$, $X_{22a}^* = d_2$.	$X_{11a}^* = 40$, $X_{21a}^* = \min\{10, d_2 - 50\}$, $X_{22a}^* = 50$, $R_2 = \max\{0, d_2 - 60\}$.	$X_{11a}^* = 40$, $X_{22a}^* = \min\{50, d_2\}$, $R_2^* = \max\{0, d_2 - 50\}$.
Solution type	'fully centralised processing'	'distributed processing'	'centralised processing by node'

Table 4.1. Solutions for the simplified PDS formulation.

As can be seen from Table 4.1, all three solution types are produced from the model, for different values of the v and d_2 parameters. In order to outline the solution process for Table 4.1 it is important to note that it is no longer profitable to meet demand if more than 50 units of processing capacity are used at each processing node. Demand is rejected after this processing capacity has been used.

In the first row of Table 4.1 (v = 5) the transportation costs are such that it is most profitable to meet demand at processing node 2, regardless of where it was received. Hence, all demand will be met at this node until 50 units of processing capacity are used, and then the remainder would be met at processing node 1. Note that when v = 5 and d_2 = 50 a 'centralised processing by node' solution occurs. In the first two columns of the second row ($5 = v = 30$) the transportation costs are such that whilst it is still profitable to meet demand if routing is required, it is most profitable to first meet demand at the node where it was received. In the final column (v = 30) the costs of transportation are such that it is not profitable to route flow in order for it to be

processed. Hence, demand is met at the node where it was received until 50 units of processing capacity are used at that node.

4.3. Investigating the Impact of Model Complexity

There are a number of possible approximations of the PDS model, particularly with respect to the delivery time functions. It is important to understand the implications, with respect to solutions produced, of making such approximations to this model. This will help identify the important operational model detail.

4.3.1. No Delivery Costs

Simplifying the model by removing real and penalty costs of routing and processing times, making the model similar to that in Tomasgard (1998), makes the solution process straightforward. With only unit processing costs remaining in the objective function, and with unrestrictive node capacities, if any processing node m with unit cost $b_m > \min_n b_n$ is being used in the optimal solution, the total cost can be reduced by moving all that processing to a cheaper processing node. Since the revenue received for the service is the same wherever the demand is processed, all demand for all services would be met at the cheapest processing node. Hence, this model with no delivery costs always produces 'fully centralised processing' solutions.

4.3.2. Constant Delivery Costs

This simplified model has constant delivery costs. A straight linear approximation is made of the delivery time function, so congestion is not considered. From the discussion in Section 4.2.2 it follows that this model will produce 'centralised processing by node' or 'fully centralised processing' solutions. Strictly 'distributed processing' solutions will not be produced as the marginal costs are constant, so the original node will remain the best option for meeting all a location's demand. Because there are unlimited capacities, considering the binary aspect of subservice installation would not change this result.

4.3.3. Processing Times Only

This simplified model has convex processing times as the only component of delivery times. Transportation times, and their associated convex functions, are removed from the model. All types of solutions can occur. This can be seen from the example in Section 4.2.3 with the arc costs set to zero, $v = 0$ (Table 4.1).

4.3.4. Transportation Times Only

This simplification has convex transportation times, modelled again as piece-wise linear, as the only component of delivery times. Processing times are removed from the model. A simple example shows that this approximated model produces all three solution types. The network is shown in Figure 4.5, other data are as before.

Let:

$X_{dr,n,j}$ = demand received at node dr that is met at processing node n where the transportation time is at rate j, where $j = a$ if the amount routed through R1 is below 50 units, and $j = b$ if the amount routed through R1 is above 50 units. Note that the j index is only required if $dr \neq n$.

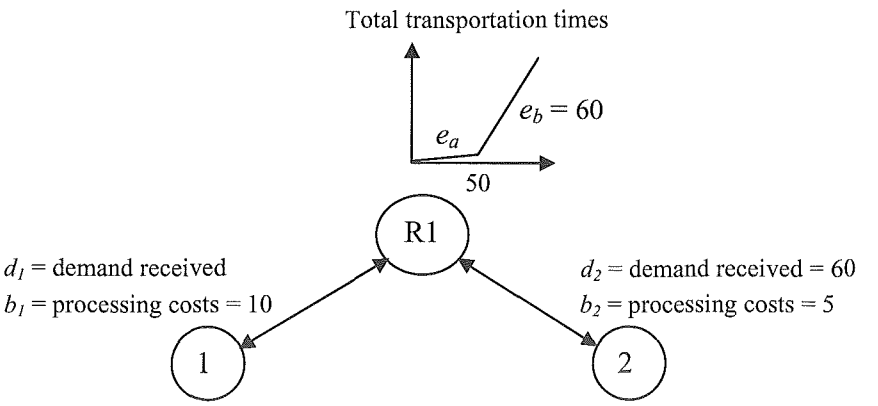


Figure 4.5. Simplified network for PDS model with transportation times only.

The resulting formulation of the PDS model for this example would produce the solutions presented in Table 4.2.

	$e_a \leq 5$		$e_a \geq 5$
	$d_1 \leq 50$	$d_1 > 50$	
Solution (non-zero variables)	$X_{12a}^* = d_1$, $X_{22}^* = 60$.	$X_{11}^* = d_1 - 50$, $X_{12a}^* = 50$, $X_{22}^* = 60$.	$X_{11}^* = d_1$, $X_{22}^* = 60$.
Solution type	'fully centralised processing'	'distributed processing'	'centralised processing by node'

Table 4.2. Solutions for the PDS model with transportation times only.

Table 4.2 shows that this example produces all three solution types. In the first two columns the transportation time costs are such that it is most profitable to meet all demand at processing node 2, regardless of where it was received. Once the flow from routing demand from node 1 exceeds 50 units, all remaining node 1 demand is met at node 1. In the final column demand is met where it is received.

4.3.5. Summary

Because congestion is a real occurrence, the delivery time function is indeed convex in reality. Models that include the convex congestion function provide a richer array of solutions than the simpler models. These suggest that the simplified models of Sections 4.3.1 and 4.3.2 are inadequate for decision-making in reality. The complexity of the convex delivery time functions appears necessary.

Note that, with these simpler models, restrictive capacities can cause some degree of distributed processing, as demand is forced to another node as capacity levels are reached. However, this distributing of processing is very dependent on the capacity levels.

4.4. Other Objective Functions

Whilst being the most accurate representation of the processing decisions objective, modelling revenue less quality of service penalties is more complex than a number of

other potential objectives. This section investigates two other potential objective functions for the PDS model.

4.4.1. Minimise Rejections

Tomasgard (1998) minimises rejections in his processing decisions model. This objective is used because, regardless of other costs, it is clear rejecting demand is harmful to the service provider's well-being. It is possible that this objective could be used in association with the 'effective capacity' constraint suggested by Le Blanc *et al* (1999), discussed in Chapter 2. The effective capacities would 'force' distributed processing in the model when appropriate. However, under the profit maximising objective the model balances whether to install additional subservices against the cost of increased delivery times of doing so. A model that solely minimises rejections would not do this.

4.4.2. Minimise Delivery Times

Because of their importance in maintaining customer satisfaction (which is critical for the service provider's long term survival) minimising delivery times, and hence the associated congestion, is another possible objective. However, since meeting any demand incurs delivery times, all demand would be rejected using this objective. This holds true unless a constraint ensuring a certain amount of demand was met was added; but this adds problems in itself, such as choosing this level of demand. This objective is inappropriate because it focuses solely on one aspect of the problem situation, delivery times. It ignores the trade-off between rejections and delivery times, discussed in Chapter 3. The best solution should balance these two factors.

4.5. Summary

The analysis conducted in this chapter on the processing decisions model presented in Chapter 3 has revealed some important information about the operational model, particularly with respect to the appropriateness of model approximations. Firstly, however, it was found that the model's solutions could be grouped into three general

solution categories, where the categories were distinguished by the extent to which a solution distributed its processing. Distributed processing occurs when marginal costs increase as the amount of demand met increases, whereas centralised processing solutions occur when marginal costs are relatively constant. An example was presented to illustrate when each solution type occurs.

Following on from the identification of solution types, Section 4.3 concluded that simplified models are inadequate for decision-making in reality. This is because these models do not produce strictly 'distributed processing' solutions, which would likely be optimal as the marginal delivery time costs increase as more demand is met due to congestion. It was concluded that the complexity of the convex delivery time functions was likely to be important for the capacity model.

Finally, the possibility of using simplified objective functions was eliminated. This is because these objectives (minimising rejections and minimising delivery times) do not properly consider the trade-off between these two factors.

5. PROCESSING CAPACITY EXPANSION DECISIONS

This chapter begins to investigate the system in which the processing capacity expansion decisions are made. This includes how the processing decisions model needs to be adapted to be appropriate in this environment.

5.1. Introduction

Processing capacity expansion decisions are infrequent large-scale strategic projects that assess the current levels of processing capacity available and determine what changes to these levels are necessary to ensure the long term viability of the firm. The capacity decisions made are crucial because, as discussed by Simampo and Ryan (2001), the service provider does not want to lose customers due to poor quality of service. At the same time, with competition driving down prices and limited

advertising revenue, profit margins are narrow. Therefore, managing their capital investment in capacity is crucial for a service provider's survival.

These expansion projects require a substantial commitment of capital resources, which will only be re-paid over a long period of time. Because of the risks and uncertainties involved with this, it is important to conduct extensive modelling and planning efforts to identify good capacity expansion decisions. The importance of these decisions and the need to analyse them carefully is emphasised in the following two examples.

- In 1996 America Online changed its pricing structure so that its customers could get unlimited Internet access for a cheap monthly fee. This change resulted in a huge demand influx (daily on-line sessions increased by one-third, and average subscribers stayed on-line 20 percent longer than before). The influx was so great that their processing ability was inadequate and, according to Lewis (1996), the majority of customers who attempted to use the service were rejected. Customers who had entered an agreement guaranteeing them Internet access were understandably unhappy and Lohr (1997) reports that law suits ensued.
- In New Zealand there have recently been a number of attempts at allowing customers to have free Internet connection (such as by Zfree, i4free and Freenet), for example, Barton (2000). However, all of these companies had to stop accepting new customers after only a short period, IRN (2000). This was because their processing capacity levels could not provide reasonable quality of service for their quickly expanded customer base.

The aim of this chapter and Chapters 6 and 8 is to investigate the problem situation and build a mathematical model for the processing capacity expansion decisions. This chapter identifies the relevant factors and relationships the decision maker should consider when making these decisions. This includes outlining the necessary changes to the processing decisions model presented in Chapter 3 to enable it to be used in the strategic planning environment. Chapter 6 goes into greater detail about one of the most important and complex factors in this system, demand. Finally, Chapter 8

presents the mathematical model formulated for the processing capacity expansion decisions to improve decision-making in this area.

5.2. Problem Situation Part I: Diagrammatic Overview

The decision is how best to invest in processing capacity in order to meet demand over the planning horizon. Figure 5.1 presents an influence diagram for the system of interest. It shows all the factors and relationships that should be considered in this decision-making environment. The dotted segment in this figure is a simplified representation of the processing decisions environment, which was outlined in Chapter 3.

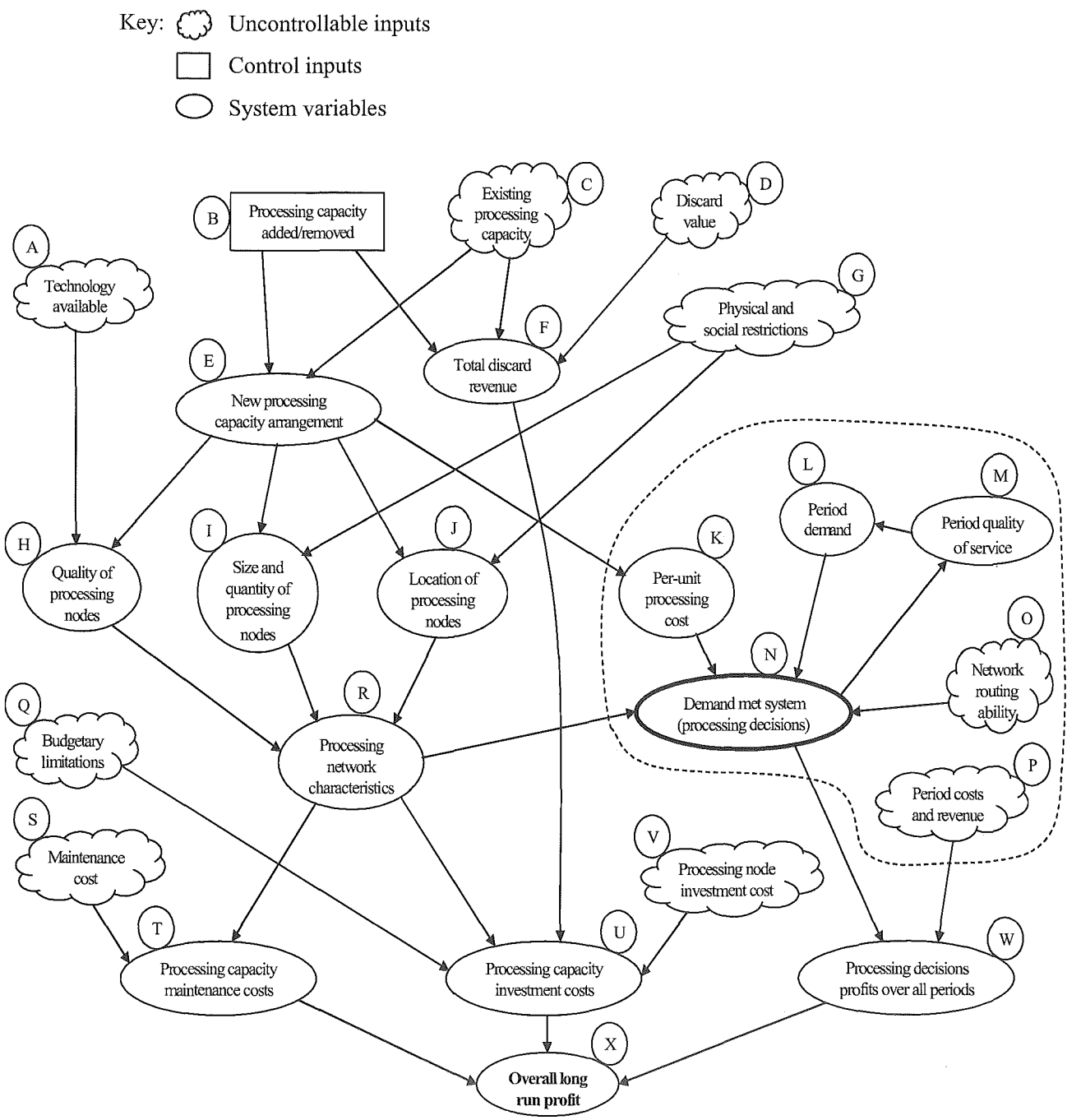


Figure 5.1. Influence diagram representing the processing capacity investment decisions system.

A brief explanation of this system, which will be explained in detail in the remainder of this chapter, is as follows. The service provider invests in processing resources, specifically setting the size, number, technology, and location of the processing nodes used in the models in Chapter 3. Costs are incurred in obtaining and maintaining these processing resources, but, by being used to meet demand, the processing resources earn the service provider revenue over time. This revenue is, however, dependent on the processing resources provided. The service provider must properly consider this trade-off between costs incurred now and future revenue.

For ease of understanding, this thesis does not consider the timing aspect of the expansion decision. Processing expansion can only occur at the beginning of the investment planning horizon. Examples of papers that do consider this extension are Aneja and Chaouch (1993) and Neebe and Rao (1986). We do, however, briefly discuss this aspect as a model extension in Chapter 10.

5.3. Problem Situation Part II: Detailed Discussion

This section provides a detailed discussion of all the factors and relationships in the system identified in the influence diagram (Figure 5.1). In the following discussion, letters appearing in brackets refer to the part of the influence diagram currently being discussed. The discussion will begin with the overall long-run profit, and then work through the costs and revenues this consists of, working roughly from left to right in the influence diagram.

5.3.1. Long-Run Profit

In order to determine the best levels of processing capacity (H, I, J) it is important to look at how to measure the quality of potential solutions. In this research, to ensure that the aforementioned trade-off between investment costs and future profits is properly considered, the planning horizon's expected long-run profits (X) are used. This profit consists of the expected long-run revenue from meeting demand (W), less the investment (U) and maintenance (T) costs. Because they are incurred immediately, the costs are easily determined. Conversely, as they are earned over

time, the revenues are uncertain. Adding even more complexity, they also depend on the capacity provided. Chapter 6 looks more into this. For discounting purposes it is assumed that processing decisions profits are earned at the end of their respective periods.

5.3.2. Maintenance Costs

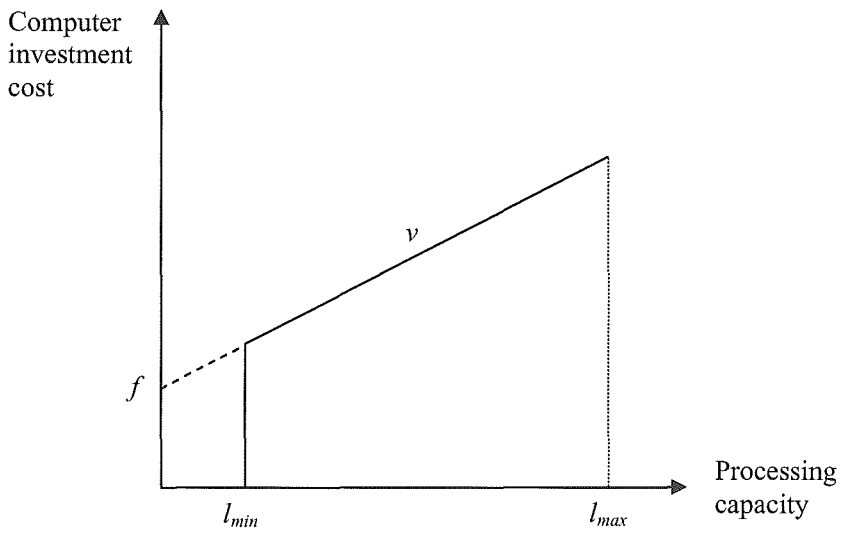
Computer maintenance incurs a cost (S), which depends on the location and technology of the processing node installed. If appropriate, the maintenance costs could be zero, such as when the service provider hires the processing capacity. This cost may also include insurance and other per-computer costs. Maintenance costs are included in the model as a fixed cost incurred for every computer in the network, regardless of how much processing capacity is installed.

5.3.3. Investment Costs

The cost of obtaining processing capacity at a potential location (V) consists of two parts. The first is the fixed cost associated with purchasing the computer on which the processing capacity will be installed. The second part is the variable cost associated with the capacity size bought. Some processing capacity may also be hired on a pay-per-usage basis. In this situation additional costs would be incurred at the time of use.

Many factors influence the fixed and variable costs. For example, if the expansion decisions are for an operating service provider, there will be existing levels of processing capacity (C). This influences the initial fixed costs because a new computer may not need to be purchased immediately, and because of the presence of discard costs / revenues (D). Existing processing capacity is, however, not explicitly considered in this model. We include variable investment costs in the first stage of the model to ensure that unused processing capacity also incurs this cost. Note that even though we do not consider routing expansion, routing links can be added to connect non-existing processing locations to the routing network, so long as the geography of the network is not altered. The cost of this is included in the fixed cost of obtaining the processing node.

This fixed plus variable investment cost structure is prevalent in the literature in similar capacity expansion situations (for example, Friedenfelds (1981), Luss (1982), Dutta and Lim (1992), Li and Tirupati (1994)). Similarly to this work, we assume the investment cost function is known for each potential computer. Figure 5.2 presents an example of the total investment costs, which are a function of the amount of processing capacity installed. The model allows upper and lower bounds on computer capacity.



f – fixed cost for computer.
v – variable cost for computer.
l_{min} – minimum size of computer.
l_{max} – maximum size of computer.

Figure 5.2. Investment cost structure for installing processing capacity.

5.3.4. Physical and Social Restrictions

Often factors outside the service provider's control (G) restrict their capacity expansion decisions. Investment costs are constrained by budgetary limitations (Q). Qualitative factors, such as political and social considerations, are not considered explicitly in the mathematical model.

5.3.5. Technological Improvements

One of the major issues when making decisions in the rapidly changing industries of telecommunications and the Internet is the technological progress that is being made. Routing technologies have improved drastically, by means such as digital technology, the use of fibre optic cables, and the adoption of ISDN. In a similar manner, processing technologies are also drastically improving both in terms of processing speed and cost.

The main advantage of newer, higher quality, processing technologies (A) (as assumed in this research at least) is that they will be faster at processing demand. Using these newer technologies would lead to reduced delivery times, and as a result, potentially greater customer satisfaction and increased profits. However, these technological improvements do come with a higher investment cost. Therefore, the service provider has to weigh up the costs of newer technology against the benefits, to determine what kind of technology to obtain. This is important because, as newer products become available, older products can become cheaper to use. Indeed there are a number of examples where it is best to use older technology even when newer technology is available (for example, Rajagolapan *et al* (1998)).

Similarly to the method used by Jaskold-Gabszewicz and Vial (1972), in modelling terms different types of technology are simply computers with different processing time characteristics (different processing time rates and breakpoints) and investment cost structures. An example is shown in Figure 5.3.

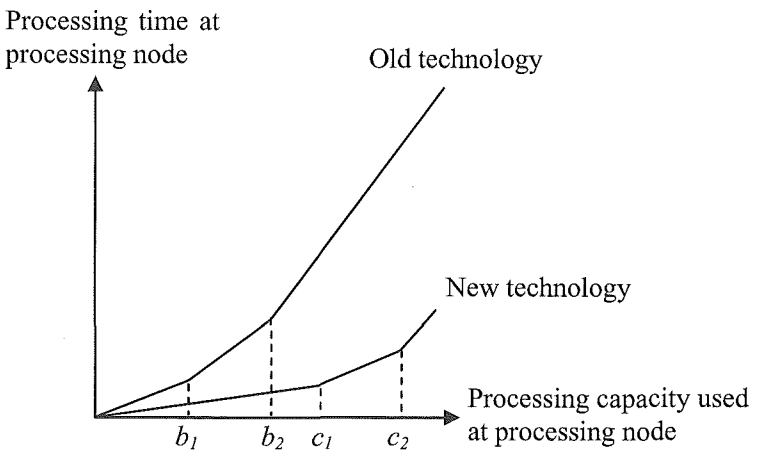


Figure 5.3. Modelling different processing technologies.

It has been assumed for this research that all computers (of all technologies) can run all subservices. This assumption was made because including different types of capacity, such as done by Rajagopalan (1994), would complicate the model rather unnecessarily. However, the model could easily be extended to include this.

5.3.6. The Processing Decisions Level

As discussed in Section 5.3.1 and in previous chapters, the processing decisions provide the only source of revenue to reimburse the investment costs. Because of this, and because this revenue is dependent on the processing resources provided, it is important that the processing decisions are considered in the capacity expansion decision-making. However, whilst it is based on it, the model used to consider the processing decisions is not the PDS model presented in Chapter 3. Subtle adjustments are made, replacing the penalties on delivery times and rejections with a different approach, included in the constraints, to model the direct effect of these considerations on demand. This approach is presented in Chapter 6. Other necessary changes are now outlined.

Processing Times

The processing time functions are as described in Chapter 3, except in the capacity expansion model the processing node size is now variable. Hence, the processing time function should vary with capacity. In order to model this it is assumed that the breakpoints of the processing time function are at the same relative point regardless of the processing node size. So, the first breakpoint in the processing time function is at b_1% of the total processing capacity, the second at b_2%, and so on for all the processing time segments. It is assumed that the processing rates and percentage breakpoints are known for each computer at each location, and these are technology-dependent.

New Services / Subservices and Locations

Services and subservices will change over the planning horizon. New services will be developed and current services and subservices upgraded. These changes are uncertain at the time the capacity expansion is made. For the model, a service is characterised by the subservices it uses. When a subservice is changed its characteristics change (such as requiring less processing capacity when installed). This can be modelled as a new subservice (with the altered characteristics). It follows that the services that used the old subservice will now use the new subservice and, hence, are new services themselves. These new services are included in the model with zero initial demand.

It is assumed that demand is only received at existing locations. This assumption was made because considering new locations would require additions to the existing routing network, which is outside the bounds of this research. Similarly, it is assumed that there are no changes to the routing network over the planning horizon, meaning the transportation time functions and arc capacities are constant over time.

Demand

Demand (L) is probably the most important factor when considering the level of processing resources to provide. How demand is met (N), and the resulting quality of

service provided (M) influences the demand over time, creating an inherent feedback loop, as discussed in Chapter 3. Modelling this feedback loop, as seen in the influence diagram (L-N-M-L), adds significant complexity to the model. Complicating matters further, the quality of service able to be provided depends on capacity, a model variable. In light of this importance and complexity, the discussion and modelling of demand is left for Chapter 6.

5.4. Summary

This chapter described the system of interest for the processing capacity investment decisions by presenting an influence diagram and then providing a detailed explanation of the factors and relationships. The inclusion of these factors in the mathematical model was discussed, and in particular, what approximations and assumptions were made. The following chapter goes into greater detail about one of these factors, demand.

6. DEMAND

Demand is a crucial factor to consider when setting processing capacity. Because of its complexity this entire chapter is devoted to it, including outlining how it was modelled.

6.1. Introduction

As illustrated by the examples at the beginning of Chapter 5, a crucial factor when determining processing capacity levels is the future demand for services. If future demand were known, then decision-making would be relatively easy, in the sense that a deterministic mathematical model could be solved to find the optimal capacity. However, decision-making in the telecommunications and Internet environment is made more difficult because demand is variable (see for example, Wirth (1997)). With uncertain demand patterns, demand must be estimated for the model. This can be done by forecasting demand exogenously, or by modelling demand endogenously using the factors that influence it. Complicating matters, some of the factors that

influence demand are model decision variables, such as capacity levels. The two methods are compared in Chapter 11.

Section 6.2 outlines the factors that influence demand, and which of these we predominantly focus our attention on. Section 6.3 discusses how demand evolves over time, and Section 6.4 describes how we model this evolutionary process. To our knowledge, this is the first in-depth discussion and modelling attempt that considers demand being variable, based on service levels.

6.2. Factors Influencing Demand

In order to model demand, we must model the consumer behaviour that influences it. From the literature (for example, So and Song (1998), Chu and Altmann (2000), and Simampo and Ryan (2001)) it is apparent that the two most important factors influencing consumer behaviour towards requesting services are price and quality of service. Hence, we only explicitly consider these two factors in our model. Anything else influencing consumer behaviour is combined in an 'external' factors parameter. Also, the model only considers influences from the immediately previous period. This assumption could be relaxed if necessary, further complicating the modelling of demand.

Therefore, a period's demand (D_{p+1}) is modelled as being influenced (independently) by the level of the demand in the previous period (D_p), price (P), quality of service, and other 'external' factors. This chapter refines this model:

$$D_{p+1} = D_p + f(\text{P, quality of service, 'external' factors}).$$

6.2.1. Price

In this thesis it is assumed that the service provider cannot influence the price they receive for a service; that is, they are a price-taker. This simplifying assumption was made in order to remove the strategic implication of markets (such as gaming). Under this assumption price and demand are independent. Price is included at its expected

value in the modelling process, avoiding the need to model the price elasticity of demand. Hence, the model becomes:

$$D_{p+1} = D_p + f(\text{quality of service, 'external' factors}).$$

See, for example, Gaimon and Ho (1994) and Kim and Lee (1998) for capacity expansion research that has looked at demand as a function of price.

6.2.2. Quality of Service

According to Graham (1991), quality of service is a measure of the difficulties a user experiences when requesting demand. In the model, these difficulties are measured by the number of customer requests not met (rejections) and the speed with which, on average, requests are met (delivery times). Anything else that could influence a customer's view of quality of service is included in 'external' factors.

Customers Expect Certain Quality of Service

The impact that the quality of service provided has on demand depends on the level of quality of service the customers expect (CE). This level of quality of service depends on a number of factors, including the quality of service customers are used to, and priorities on their time. New technologies also create an expectation of better service, regardless of whether the service provider has this new technology. It is difficult to judge how these factors influence the quality of service the customers expect. Therefore, this factor is modelled as an uncertain parameter. This parameter's distribution could be estimated from customer behaviour surveys or analyses (such as Chu and Altmann (2000)) and is dependent on the particular problem situation. Note that each customer is likely to have a different opinion of quality of service. In the model the aggregate level of quality of service is used:

$$D_{p+1} = D_p + f(\text{quality of service, CE, 'external' factors}).$$

For now, parameter CE, and the other parameters identified as being stochastic, will be thought of deterministically so as to not complicate the understanding of the model. The parameters' stochasticity will be considered in Chapter 8.

Quality of Service Influencing Demand

As discussed by Evans (1997) and Taylor (1994), demand will increase or decrease based on how the quality of service provided compares with what the customers expect. If the customers view the quality of service received as being poor, then there will likely be a reduction in demand in the next period. This reduction occurs because customers become dissatisfied and reduce their own level of requests, as well as reducing the number of requests from other customers due to word of mouth. Providing a quality of service that closely matches what customers expect is assumed to have no impact on the demand in the next period. Customers will have expectations on both aspects of quality of service (rejections and delivery times), and both aspects are compared with their expectations independently. CE describes both these two aspects.

Note that as demand changes due to quality of service, so will the average level of quality of service customers expect. This is because if quality of service is such that demand decreases, those customers with the greatest expectations of quality of service will be the ones that have left. As a result the average level of quality of service expected by the remaining customers will not be as high. This is not considered explicitly in the model.

Processing Capacity

The quality of service with which the service provider is able to meet demand is dependent on the processing resources available (C). If more processing resources are provided there will be less processing congestion and higher quality of service[7]. Therefore, the processing capacity influences demand through the quality of service provided.

Hence, quality of service (QoS) is measured by the total number of rejections and the average delivery time (total delivery times divided by the number of demand requests

[7] Note that the network topology also influences the level of quality of service.

met), and is dependent on the processing capacity provided and the number of demand requests met. Because of this the model becomes:

$$D_{p+1} = D_p + f(QoS_p(D_p, C), CE, \text{'external' factors}).$$

The above function for next period's demand shows the feedback loop referred to in Chapters 3 and 5, where a period's demand influences quality of service, which influences the next period's demand, and so on. Note that using aggregate measures of quality of service like this is common in communication industries (for example, Gavish (1992)).

6.2.3. 'External' Factors

External factors are those that either the decision maker has no control over or are outside the scope of this model. Examples include the state of the economy and customers preferences when using the Internet or telecommunications for their services. Also, we do not explicitly consider the reliability of network components, which can influence the quality of service able to be provided. This aspect is implicitly included in the changes to demand due to external factors. For an examination of network reliability and its impact, see Courcoubetis *et al* (2000), for example.

In the model the change in demand due to external factors is included as a random movement (?) up or down, which has a known distribution. The variance of this distribution gives an idea of demand volatility. Hence,

$$D_{p+1} = D_p + f(QoS_p(D_p, C), CE, ?).$$

6.2.4. Maximum Demand Level

The changes to demand due to quality of service will be bounded by a maximum level, above which the demand potential is exhausted, and no more customers will request services, no matter how good the quality of service. This maximum demand parameter is modelled as stochastic, with a known distribution based on, for example, the total number of Internet / telephone users.

6.3. What Happens to Demand Over Time?

6.3.1. Demand Can Fluctuate

Once the processing resources are fixed, the quality of service provided depends on the demand met. When there are few customers there will be little congestion, and these customers will receive a quality of service that is above what they expect (such as b_0 in Figure 6.1), increasing demand[8] in the next period (the next period's demand is to the right of b_0, towards b_1). Meeting the increased demand adds to the level of network congestion, decreasing the quality of service provided. If the demand increases to a level that the quality of service provided is worse than what customers expect (such as b_2 in Figure 6.1), demand would decrease (the next period's demand is to the left of b_2, towards b_1). The remaining demand can now again be met with a better quality of service. The closer to b_1 the current demand is the smaller the resulting change in demand (as quality of service is not too far from what customers expect).

$$QoS_p \; ? \quad D_{p+1} \; ? \quad QoS_{p+1} \; ? \quad D_{p+2} \; ? \quad \ldots$$

As illustrated in Figure 6.2, because of this process demand will settle near b_1 (LD), where the quality of service provided is close to the level customers expect[9]. If external factors cause demand to move away from its long-run level (> or < b_1), demand will still gravitate back to LD because of the process discussed above. As happened in the case described by Lohr (1997), we assume demand settles quickly, within 10-20% of the duration of the planning horizon, so as to make it unnecessary to consider the level of demand during the settling period. Hence,

$$LD = \lim_{p \to \infty} D_p + f(QoS_{p-1}(D_{p-1}, C), CE).$$

---

[8] 'Demand' refers to the service provider's demand, not overall Internet demand.

[9] Because of time lags, and depending on how it's modelled, demand could oscillate around LD. However, the average long-run demand remains the same.

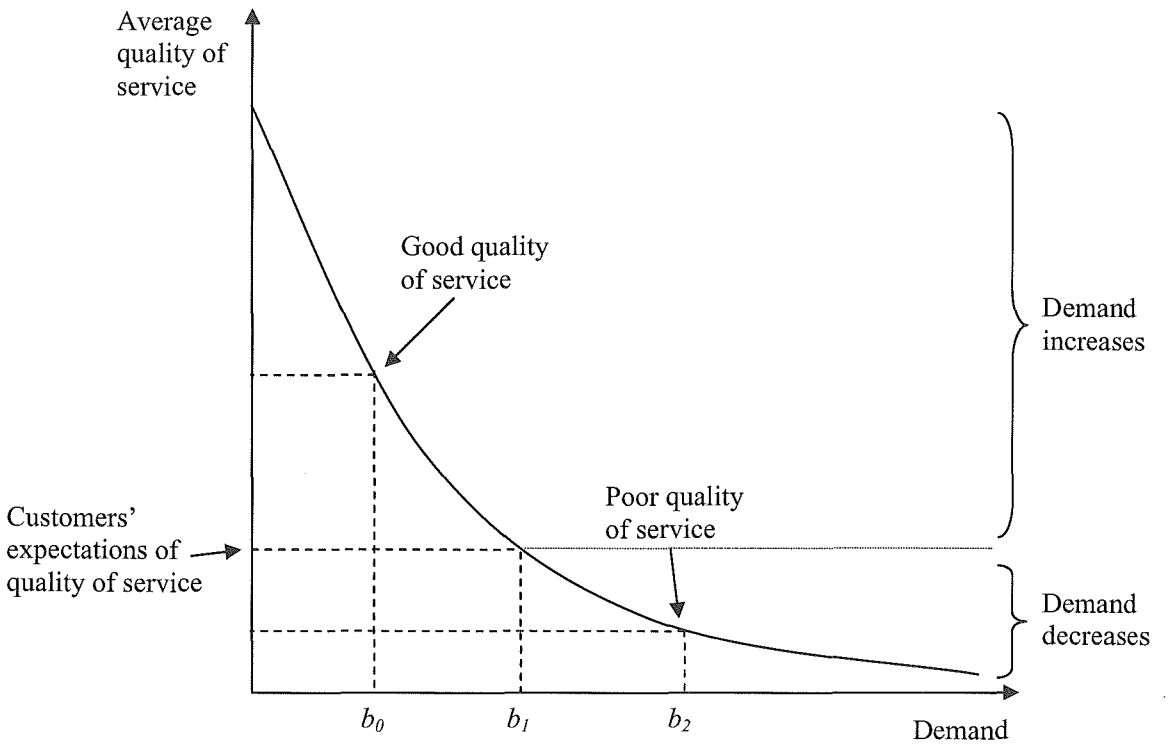


Figure 6.1. Quality of service relative to what customers expect versus demand.

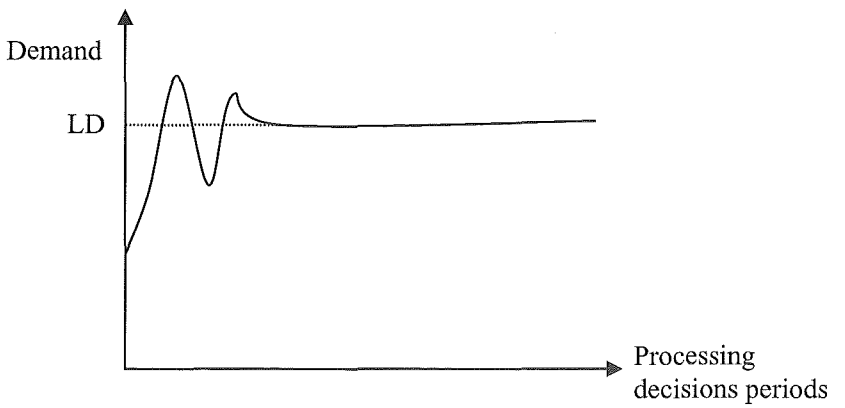


Figure 6.2. Demand over the planning horizon for a fixed processing capacity.

6.4. The Modelling of Demand

As outlined in the previous section, over the planning horizon demand can fluctuate from period to period but will quickly settle at a long-run level. Because of this process we model demand at its long-run level, LD. However, in order to validate this approximation we developed a model, presented in Chapter 7, that simulates how demand might change from period to period, dependent on the previously-identified factors. Testing performed using this model (presented in Appendix A) shows that long-run demand is a reasonable approximation of average demand (which is a good estimator of long-run revenue) under high and low demand volatility.

This section looks at the modelling of demand at the long-run level. First, appropriate assumptions are made.

6.4.1. Demand Assumptions

Removing Period to Period Dynamics

Sethi *et al* (1995) conclude that the capacity decision can be made under aggregated information from the operational level, by including these parameters at their expected values in the model. This implies that the period to period dynamics regarding demand are unimportant, and using average or peak demand estimates is appropriate. Using average demand is inappropriate because the resulting capacity levels would be inadequate for meeting peak demand. Hence, peak demand is used. Off-peak demand periods are excluded, because any solution able to meet peak demand with a satisfactory quality of service will be able to easily meet off-peak demand, and because peak demand periods are likely to provide a greater proportion of the service provider's overall profit.

Second stage parameters other than demand are hence included at their expected values or stochastically, where the weightings of scenarios depend on the proportion of time the periods are in that state and the probability of the scenario occurring.

6.4.2. Modelling Long-Run Demand

To model demand its long-run level, LD, must be determined. LD is constrained by the following factors. First, at the equilibrium level the average quality of service cannot be worse than the level the customers expect. Second, demand cannot exceed its maximum level. Hence, given the optimisation environment, long-run demand will be the highest level of demand that satisfies the above two constraints. Precise details of the how demand is modelled at its equilibrium long-run level are outlined in Chapter 8.

Modelling demand in this manner means it is now unnecessary to use delivery time and rejection costs to discourage poor quality of service (as seen in the processing decisions model formulated in Chapter 3). This aspect is now covered by constraints that ensure acceptable quality of service. These constraints were not used in the PDS model because at the operational level demand must be met or rejected as is best for that period. Long-run principles, such as is implied by using constraints, are not appropriate in that model's time-frame.

This philosophy of variable demand is the same as alluded to by Gartner and Nelson (1998):

> *"Carriers can no longer afford to forecast demand first and then build just enough capacity into their networks to hit that projection. Forecasts fall short of capacity needs, not to mention user expectations... When carriers build a network with excess capacity, the customers will come."*

We agree that capacity dynamics can influence the quality of service provided, and thus a service provider's demand; our model for endogenous demand outlined in this chapter recognises this. However, we realise demand will be finite by considering a maximum demand level.

Implications of Modelling Long-Run Demand

Dramatic simplifications to the model occur when demand is modelled at its constant steady-state. The periods become independent of one another, and because the dynamics are removed, with parameters at their expected values or considered stochastically, each processing decisions period is now making the same decisions under the same set of parameters. This enables the model to be re-formulated as consisting of only one second stage period, with its profit multiplied n times to reflect the number of periods in the planning horizon. Simplifying the multi-stage element also better illustrates the features of the model.

6.5. Spread of Demand

The modelling of demand outlined in the previous sections finds a period's overall long-run demand (LD). However, the service provider will want to know more accurate information than this for decision-making, such as how that demand is spread around locations and services. For each service (s) received from each location (dr) the same logic of long-run demand applies ($LD_{s,dr}$). In the model, $LD_{s,dr}$ is found as a fraction of LD. The value of the fraction for a given service received from a given location will depend on many things, such as the size of the location, and the popularity and expected life of the service. Because of the uncertainty over this, and the more volatile nature of $LD_{s,dr}$, these fractions are modelled as stochastic. This is discussed in more detail in Chapter 8.

An alternative to comparing quality of service over all customers could be to compare the average quality of service provided for the group of customers requesting a particular service from a particular location with what those customers expect. However, this information cannot be determined precisely from the model, and the best approximation is non-linear.

Groups of customers requesting a particular service (s) from a particular location (dr) share processing and routing nodes with customers from other service / location groups. Hence, to be able to compare the quality of service provided with what the customers expect at this level, the total processing or routing time at each node (n)

must be shared appropriately between the groups that use the node. This could be approximated by finding the average delivery time for each node and multiplying this by the node usage of each group, as shown in the following term. This calculation is non-linear, as even when $dt_{s,dr,p}$ is constant it involves multiplication and division of model variables.

$$dt_{s,dr,p} = \frac{\text{Node usage}_{s,dr,p}}{\text{Total node usage}_{n,p}} \text{Total time}_{n,p}$$

6.6. Summary

Demand is a crucial factor in the capacity expansion model. This chapter has proposed and developed a model of demand over the time horizon. Demand is modelled as being influenced by price, quality of service, and external factors, where most effort was spent modelling the impact quality of service has on demand. This is because if the level of quality of service the customers expect is not met, demand is likely to decrease. The feedback loop this creates, where a period's demand influences the period's quality of service, which influences the next period's demand, and so on, creates significant modelling complexity. Demand can fluctuate because of this (as discussed with respect to Figure 6.1), but settles at a long-run level, at which it is modelled in this chapter. Importantly, the spread of demand was also considered. Chapter 8 uses this modelling approach to develop mathematical models for the processing capacity investment decisions. However, before this, Chapter 7 presents a model that simulates period to period changes in demand.

7. SIMULATING DEMAND

This chapter develops a model that simulates how demand changes from period to period depending on the quality of service provided and external factors. This model is used to validate the modelling of demand at its long-run level, as well as being used in Chapter 11 to evaluate expected long-run profits of capacity solutions.

7.1. Introduction

This chapter outlines a model developed to simulate how demand changes from period to period given fixed capacity (RP model). Section 7.2 outlines how the factors identified in Chapter 6 as having an influence on demand are modelled in this simulation. Parameters necessary for this model are defined in this section and are summarised in Table 7.1. Section 7.3 then presents the simulation model, specifically outlining how the model iterates over the many processing decisions periods in the capacity planning horizon, as well as detailing precisely how demand changes from period to period due to the factors discussed in Section 7.2.

7.2. How Factors Influence Period to Period Demand

7.2.1. Rejections

The reduction in future demand due to rejections in a period (R_p) is modelled as a fraction of the amount of demand rejected. This fraction (*fr_rejs(?)*) represents the average proportion of customers lost due to rejections. Note that parameters marked with *(?)* are uncertain, but are thought of deterministically in this chapter for ease of understanding.

$$D_{p+1} = D_p - fr\_rejs\ R_p$$

7.2.2. Delivery Times

As with rejections, providing customers with a particular speed of delivery influences whether that customer and others continue to request demand. The difference between the delivery times provided (av_dt_p) and the delivery times customers expect (*ce(?)*) is referred to as the 'expectations gap'[10]. A model similar to Figure 7.1 is used to represent changes in demand due to the expectations gap. This figure illustrates a number of important points. Firstly, if the service provider provides a speed that is quite similar to what the customer expects there is likely to be no change in demand (between $-ch_2$ and ch_3 in Figure 7.1). Secondly, there is a maximum possible fractional change in one period. The maximum percentage increase ($fr\_dts_1(?)\%$) and decrease ($-fr\_dts_4(?)\%$) might differ.

---

[10] A positive 'expectations gap' indicates that the delivery times were slower than the customer expected.

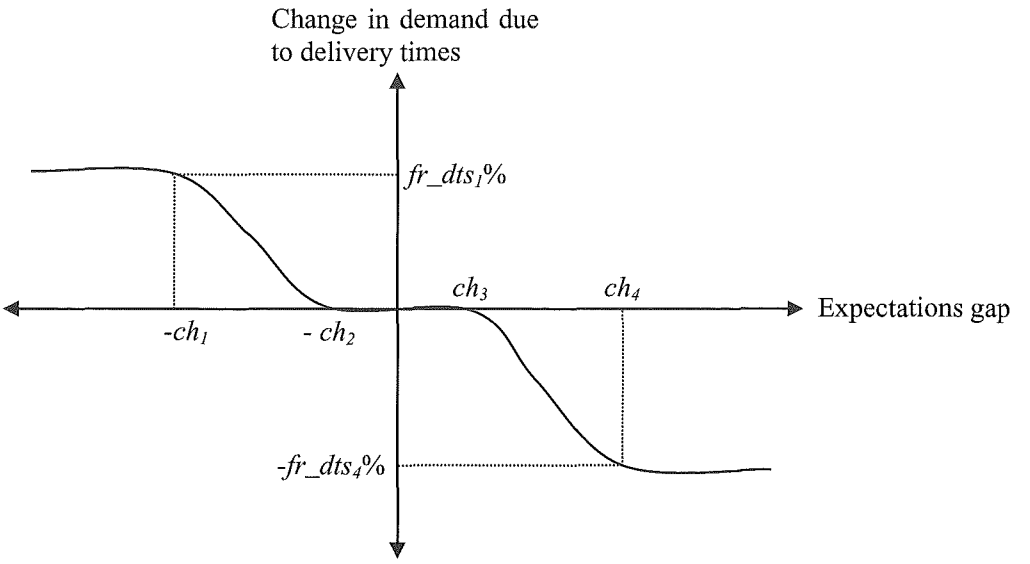


Figure 7.1. Function for demand changes due to the delivery times provided compared with the delivery times expected.

Between $-ch_1$ and $-ch_2$, and ch_3 and ch_4, the change is modelled as a step-wise function. Figure 7.2 gives an example. Demand changes due to delivery times are modelled using the same fractional change method as for rejections, where the change to demand ($fr\_dts_{nd(?)}$) depends on what 'step' (nd) of the function the expectations gap is on.

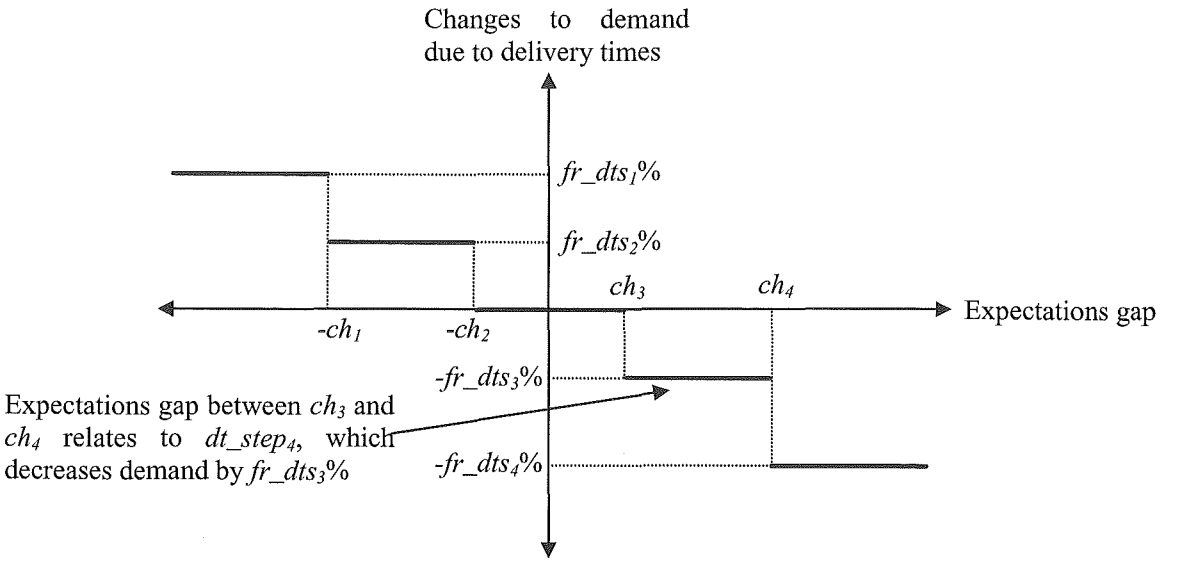


Figure 7.2. Step-wise function for demand changes due to the delivery times provided compared with the delivery times expected.

Note that it is assumed that the quality of service provided in a period only influences the demand received in the next period. Any future ramifications to demand are not considered. This assumption could be relaxed if necessary, further complicating the modelling of demand.

7.2.3. 'External' Factors

External factors ($ext(?)_p$) are included as an absolute change, independent of the current demand level. The variance of this distribution reflects the volatility of demand. External factors can cause demand to exceed its maximum level.

7.2.4. Maximum Demand Level

The demand changes from period to period due to quality of service cannot make next period's demand greater than the maximum demand (*MDP*) or less than zero. If the calculation puts a period's demand outside these boundaries it is adjusted accordingly.

7.2.5. Parameter Definitions

The parameters defined in this section are summarised in Table 7.1.

Indices and Sets

nd = number of 'steps' in the step-wise function for demand change due to delivery times.

p = processing decisions period.

Parameters

$ce(?)$ = average level of delivery times customers expect.

$ext(?)_p$ = change to demand in period p due to external factors.

$fr\_rejs(?)_p$ = the fraction of demand lost due to rejections in period p.

$fr\_dts(?)_{nd,p}$ = the fractional change to demand due to delivery times in period p if expectations gap is on step nd.

ch_{nd} = the value of the expectations gap where the fractional change to the demand goes from the ndth to nd+1th step.

MDP = maximum demand pool.

Decision variables

av_dt_p = the average delivery times received by customers.

$dt\_step_{nd,p}$ = 1, if the expectations gap in period p is on step nd, 0 otherwise.

D_p = demand received in period p.

R_p = demand rejected in period p.

Table 7.1. Indices, sets, parameters, and decision variables for the RP model.

7.3. Demand Simulation Model (RP model)

This section first provides an overview of how the demand simulation model (the RP model) iterates from period to period over the capacity model's planning horizon. Second, this section explicitly shows how demand changes from one period to the next due to the factors described in the Section 7.2.

RP Model Overview

For a fixed capacity:

Loop {p}

Solve PDS model with given demand (this model is presented in Chapter 3).

Output processing decisions profit, the number of rejections, and the expectations gap.

Update next period's demand using the process detailed explicitly below, where demand is dependent on rejections, delivery times, external factors, and maximum demand.

End Loop.

Period to Period Demand Changes

Calculating Average Delivery Times:

$$av\_dt_p = \frac{P+T}{\sum_{s,dr} Y_{s,dr}}$$ (where P, T, and Y are as defined in Chapter 3)

The average delivery time for all customers is the total delivery times divided by the number of demand requests met. This is used to determine the percentage change in demand due to delivery times from a step-wise function like Figure 7.2 (by setting the appropriate $dt\_step_{nd,p}$ to one).

Demand Changes due to Rejections and Delivery Times:

$$D_{p+1} = D_p - fr\_rejs(?)_p \, R_p + \sum_{nd} fr\_dts(\xi)_{nd,p} \; dt\_step_{nd,p} \; D_p$$

Demand in the following period equals the current period's demand less the number of customers lost due to rejections plus the change to demand due to how the quality of service provided compares with what the customers expect.

Allow for Maximum Demand Pool:

If $D_p < 0$ Then $D_p = 0$.

If $D_p > MDP$ Then $D_p = MDP$.

Updating demand based on the quality of service provided cannot cause demand to exceed its maximum level or fall below zero.

Demand Changes due to External Factors:

$$D_{p+1} = D_p + ext(?)_p$$

Finally, next period's demand is updated for changes due to external factors. These changes can cause demand to exceed its maximum level.

7.4. Summary

Demand is modelled in this thesis at its long-run level. In order to validate this modelling approach it was important to develop a model that simulates how demand might change from period to period over the planning horizon. This simulation model, developed in this chapter, is also used to estimate expected long-run profits for capacity solutions in Chapter 11.

8. PROCESSING CAPACITY MATHEMATICAL MODEL

This chapter formulates a two-stage stochastic integer program for the processing capacity expansion decisions. Included is a discussion of model stochasticity.

8.1. Introduction

Chapters 5 and 6 defined the problem situation for the processing capacity expansion decisions, and discussed how the various factors in the problem situation were to be included in the mathematical model. This chapter formulates the mathematical model.

The model has two stages, where the first stage sets the processing resources to make available for meeting demand at the second stage. The second stage represents all processing decisions periods in the planning horizon. A number of factors in the

problem situation were identified as being stochastic, that is, their values are uncertain at the time of the first stage decision-making. Hence, the mathematical model formulated is a two-stage stochastic integer program, a simplistic outline of which is seen in Figure 8.1. Before the detailed formulation is presented in Section 8.3, it is important to discuss stochastic models in general.

1st stage:

Max Expected long-run profit: processing decisions profits less the costs of providing the processing resources.

s.t. First stage constraints.

2nd stage (with uncertain parameters):

Max Processing decisions profits.

s.t. Second stage constraints.

Figure 8.1. A simple outline of the processing capacity investment decisions model.

8.2. Stochastic Models

> *"Strategic planning [such as capacity expansion] and uncertainty are twins. The first makes no sense without the latter. It is the very existence of uncertainty that forces us to produce strategic plans."*

As can be seen from the above quote from Wallace (1998), it is imperative that stochasticity is considered in the capacity expansion problem. In order to do this stochastic parameters are often characterised by continuous probability distributions. One way to make such problems tractable is to approximate the continuous distribution with a discrete distribution (Kall and Wallace (1994)). The more scenarios in this discrete distribution the better the representation of the continuous distribution, and the better the solutions to the discrete model approximate the solutions to the continuous model. However, as the number of scenarios rises, so does

the model's complexity. Hence, it is important to balance the need to keep the number of scenarios low, with the need to keep the quality of the discrete approximation high. Finding scenarios is done either by sampling or construction[11]. How scenarios are generated for the stochastic parameters in this thesis is discussed in Chapter 11. For a more general discussion of stochastic models see Kall and Wallace (1994), and of the scenario approach see Brauers and Weber (1988).

Including stochasticity means that the processing capacity expansion model has both integer decisions and stochastic parameters. Although the area of stochastic integer programming is relatively new, there has been some research in the area. The reader is referred to Stougie and *van der* Vlerk (1997) and *van der* Vlerk (2002) for a survey of stochastic integer programming and Dyer and Stougie (2002) for a discussion of these model's complexities. Also, Tomasgard (1998) discusses various solution methods for stochastic integer programs. There have been some similar two-stage stochastic integer programs in the literature. For example, Bienstock and Shapiro (1988) model resource acquisition and operational decisions, and Louveaux (1986) presents models for facility location with comparable first and second stage decisions. The key difference between these problems and that investigated in this thesis is the complex relationship between the capacity provided, quality of service, and demand.

8.3. Mathematical Model (PC1 model)

The first stage decisions are what processing capacity to install, in terms of location, technology, and amount. The second stage decisions are: what subservices to install, what demand to meet and reject, where to meet demand, and what routes to take doing so. These decisions are made after the processing capacity is fixed and uncertainty is resolved.

An exhaustive list of the indices and sets for the model are presented in Table 8.1, the parameters (lower case) in Table 8.2, and the decision variables (upper case) in Table 8.3. The constraints and objective function for the problem are then outlined.

---

[11] Construction refers to generating scenarios to closely represent the distribution and its moments.

The constraints are discussed in two sections, the first outlines the general model constraints, and the second focuses on how long-run demand is modelled. It is important to examine this latter aspect closely because this is a major modelling contribution of this thesis. The entire formulation is presented in Figure 8.2. All potentially stochastic parameters are indicated in the model (using *?*).

Indices

a = arc.

c = computer.

dr = location where demand is received.

k = subservice.

n = potential processing location.

r = route.

rn = routing node.

s = service.

Sets

J_a = routes using arc a.

K_{rn} = routes through routing node rn.

L_k = services that use subservice k.

$R_{dr,n}$ = routes between location dr and processing node n.

Table 8.1. Indices and sets for the processing capacity investment decisions model

Parameters

$b_{c,n}$ = the pay-per-usage unit cost of using computer c at processing location n.

$ß$ = the 'conversion' factor for the processing decisions profits, taking into account all periods over the planning horizon and discounting.

$ce(?)$ = the average level of delivery times customers expect (uncertain).

$d(?)$ = the maximum total grouped demand pool (uncertain).

$e_{c,n}$ = the existing processing capacity of computer c at processing location n.

$fr_{s,dr}(?)$ = the fraction of the total demand pool representing demand for service s received from location dr (uncertain).

$g_{c,n}$ = discounted cost of maintaining computer c at processing location n for the planning horizon.

$h_{k,s}$ = the processing units of subservice k used by a request for service s.

j_k = the number of units of flow required to route one unit of subservice k.

m = a large number.

$p_{s,dr}$ = the price obtained for meeting a request for service s received from location dr.

q_a = the capacity of arc a.

$re(?)$ = the proportion of rejections customers expect (uncertain).

t_a = the time it takes to send a unit of flow over arc a.

u_k = the amount of processing capacity it takes to install subservice k.

v_a = the pay-per-usage unit cost of routing a unit of flow over arc a.

w_{rn} = the pay-per-usage unit cost of routing a unit of flow through routing node rn.

z = budgeted amount allocated for processing capacity investments.

Table 8.2. Parameters for the processing capacity investment decisions model.

Decision variables

$D_{s,dr}$ = long-run demand for service s received at demand location dr.

$G_{c,n}$ = units of processing capacity of computer c added to processing location n.

$K_{c,n}$ = 1, if discarding the existing processing capacity of computer c at location n, 0 otherwise.

LD = total grouped long-run demand pool.

P = sum total of times taken to process all subservices.

$R_{s,dr}$ = units of demand rejected for service s received at demand location dr.

T = sum total of times taken transporting all the flow.

$W_{dr,n,r}$ = flow between location dr and processing node n routed over route r between those two locations.

$X_{c,k,dr,n}$ = processing units used by subservice k for meeting demand originating from location dr which is processed using computer c at location n.

$Y_{s,dr}$ = units of demand for service s received at location dr that are actually met.

$Z_{c,k,n}$ = 1, if subservice k is installed on computer c at processing location n, 0 otherwise.

Table 8.3. Decision variables for the processing capacity investment decisions model.

General Constraints

Budget Constraint:

$$\sum_{c}\sum_{n} C_{c,n}(G_{c,n}) \leq z \qquad (1.1)$$

This constraint ensures that the total investment cost does not exceed the budget. The total investment cost function ($C_{c,n}$), consisting of fixed and variable costs, is that described in Chapter 5. Binary variables are required to incorporate the fixed cost aspect of this function. This model only considers capacity expansion, but this could

easily be generalised to also consider capacity reduction by making $G_{c,n}$ unrestricted-in-sign and including the net profit of capacity reduction in the $C_{c,n}(G_{c,n})$ function. To consider existing capacity, $e_{c,n}(1-K_{c,n})$ is simply added to $G_{c,n}$ wherever $G_{c,n}$ appears in the model, and the impact of the existing processing capacity is considered in $C_{c,n}(G_{c,n})$. Finally, capacity expansion is continuous.

Demand Constraint:

$$Y_{s,dr} + R_{s,dr} = D_{s,dr} \qquad \forall\, s,dr, \quad (2.1)^{12}$$

Constraint (2.1) ensures that demand for service s received at location dr is either met or rejected.

Processing Translation Constraint:

$$\sum_{s \in L_k} h_{k,s}\, Y_{s,dr} = \sum_{c}\sum_{n} X_{c,k,dr,n} \qquad \forall\, k,dr, \quad (2.2)$$

The left hand side of Constraint (2.2) is the total processing requirement for subservice k, coming from all services that use that subservice, and the right hand side is the total processing for subservice k performed in the network.

Subservice Installation Constraint:

$$X_{c,k,dr,n} \leq Z_{c,k,n}\, m \qquad \forall\, c,k,dr,n, \quad (2.3)$$

Constraint (2.3) ensures that demand for subservice k using computer c at processing location n, can only be met if the necessary subservice is installed there. Note, for solution purposes, it is important to ensure m is as small as possible, without further limiting $X_{c,k,dr,n}$.

---

[12] Constraint (2.1) refers to the first constraint in the second stage.

Processing Node Capacity Constraint:

$$\sum_{k}\sum_{dr} X_{c,k,dr,n} + \sum_{k} Z_{c,k,n}\, u_k \;\le\; G_{c,n} \qquad \forall\, c,n, \qquad (2.4)$$

Constraint (2.4) ensures that the total processing capacity used of computer c at processing location n does not exceed the capacity of that computer. Capacity is used installing subservices and meeting demand.

Flow Requirement Constraint:

$$\sum_{c}\sum_{k} j_k\, X_{c,k,dr,n} \;=\; \sum_{r \in R_{dr,n}} W_{dr,n,r} \qquad \forall\, dr,n, \qquad (2.5)$$

This constraint ensures that the flow resulting from demand received at location dr and met at processing node n is routed between those two points.

Arc Capacity Constraint:

$$\sum_{dr,n}\sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \;\le\; q_a \qquad \forall\, a, \qquad (2.6)$$

Constraint (2.6) ensures that the level of flow on an arc does not exceed the capacity of that arc. The left-hand side defines the level of flow over an arc as being the sum of the flow over routes containing that arc. This constraint is relaxed in all test examples in Chapter 11, but is included here for completeness.

Processing Times Calculation:

$$P \;=\; \sum_{c}\sum_{n} F_{c,n}\Big(G_{c,n}, \sum_{k}\sum_{dr} X_{c,k,dr,n}, \sum_{k} Z_{c,k,n}\, u_k, \xi\Big) \qquad (2.7)$$

The piece-wise processing time function for computer c at location n ($F_{c,n}$), described in Chapter 3, depends on the amount of processing capacity available and the amount being used. Different computers and different locations can have different processing rates and breakpoints (different $F_{c,n}$). The total combined processing times are found by summing the processing times for the respective processing nodes and computers. As discussed in Chapter 3, binary variables are required to ensure that the impact on the processing times of installing subservices is correctly considered.

Transportation Times Calculation:

$$T = \sum_{rn} H_{rn}\left(\sum_{dr,n} \sum_{r \in R_{dr,n}: r \in K_{rn}} W_{dr,n,r}, \xi\right) + \sum_{a} t_a \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \qquad (2.8)$$

This constraint calculates the total transportation times. As in Chapter 3, this consists of the time spent routing through nodes and over arcs. See Appendix C for an explicit description of the piece-wise linear transportation time function.

Constraints Modelling Long-Run Demand

This section outlines exactly how the model determines LD as an endogenous variable.

Meeting Quality of Service Customers Expect:

$$P + T \leq ce(\xi) \sum_{s} \sum_{dr} Y_{s,dr} \qquad (2.9)$$

$$\sum_{s} \sum_{dr} R_{s,dr} \leq re(\xi)\, LD \qquad (2.10)$$

Considering Maximum Demand Pool:

$$LD \leq d(\xi) \qquad (2.11)$$

As discussed in Chapter 6, at the long-run demand equilibrium level the average quality of service cannot be worse than the level the customers expect. Specifically, this means the average delivery times (total delivery times divided by total demand met) cannot exceed what customers expect of delivery times (represented by Constraint (2.9)), and that the number of actual rejections cannot exceed an allowable fraction of the total demand (Constraint (2.10)). Second, long-run demand cannot exceed its maximum possible level. Constraint (2.11) ensures this. In this optimisation environment, LD will be the highest level of demand that satisfies Constraints (2.9) to (2.11).

Calculating Demand:

$$D_{s,dr} = fr_{s,dr}(\xi)\, LD \qquad \forall\, s,dr, \quad (2.12)$$

Given that these constraints determine the long-run total demand pool, Constraint (2.12) calculates the long-run demand received for service *s* from location *dr*, where this is based on the fraction of their associated demand pool that will request demand in the peak processing decisions periods.

Objective Function

$$Max \qquad \beta\, Q\,(G_{c,n}) - \sum_{c}\sum_{n} C_{c,n}(G_{c,n}) - \sum_{c}\sum_{n:G_{c,n}>0} g_{c,n} \qquad (1.0)$$

At the strategic level the objective is to maximise the profits over the entire planning horizon. This overall profit consists of three parts. First is the expected discounted return obtained from the processing decisions level for all the processing decisions periods. Subtracted from this return are the costs of providing that processing capacity: the cost of obtaining additional computers and processing capacity, and the cost of maintaining all these computers. These costs are represented by the second and third terms respectively. Binary variables are necessary to implement the conditional sum in the third term.

With $Q\,(G_{c,n}) = E(q(G_{c,n},\xi))$,

$q\,(G_{c,n},\xi) =$

$$Max \qquad \sum_{s}\sum_{dr} Y_{s,dr}\, p_{s,dr} - \sum_{c}\sum_{n} b_{c,n}\Big(\sum_{k}\sum_{dr} X_{c,k,dr,n} + \sum_{k} u_k Z_{c,k,n}\Big)$$

$$- \sum_{a} v_a \sum_{dr,n} \sum_{r\in R_{dr,n}: r\in J_a} W_{dr,n,r} - \sum_{rn} w_{rn} \sum_{dr,n} \sum_{r\in R_{dr,n}: r\in K_{rn}} W_{dr,n,r} \qquad (2.0)$$

Objective (2.0) represents the processing decisions period profits. This function is similar to the objective function for the processing decisions model (Chapter 3), maximising returns from meeting service requests less the pay-per-usage processing and routing costs. Delivery times and rejections are, however, no longer penalised in

the objective function, as this aspect is considered in the constraints ((2.9) and (2.10)). The expectation would be taken over different scenarios of the uncertain parameter(s).

The second stage decisions are made subject to the constraints of the second stage, where the values of the first stage variables are fixed, and the uncertainty is revealed.

Formulation

First stage problem:

$$\textit{Max} \qquad \beta\, Q\,(G_{c,n}) - \sum_{c}\sum_{n} C_{c,n}(G_{c,n}) - \sum_{c}\sum_{n:G_{c,n}>0} g_{c,n} \qquad (1.0)$$

s.t. Constraints

$$\sum_{c}\sum_{n} C_{c,n}(G_{c,n}) \le z \qquad (1.1)$$

$$G_{c,n} \ge 0,$$

where $Q\,(G_{c,n}) = E(q(G_{c,n},\xi))$, and the expectation is over the stochastic parameters (referenced by *?*).

Second stage problem:

$$q\,(G_{c,n},\xi) =$$

$$\textit{Max} \qquad \sum_{s}\sum_{dr} Y_{s,dr}\, p_{s,dr} - \sum_{c}\sum_{n} b_{c,n}\Big(\sum_{k}\sum_{dr} X_{c,k,dr,n} + \sum_{k} u_k Z_{c,k,n}\Big)$$

$$- \sum_{a} v_a \sum_{dr,n}\sum_{r\in R_{dr,n}: r\in J_a} W_{dr,n,r} - \sum_{rn} w_{rn} \sum_{dr,n}\sum_{r\in R_{dr,n}: r\in K_{rn}} W_{dr,n,r} \qquad (2.0)$$

s.t. Constraints

$$Y_{s,dr} + R_{s,dr} = D_{s,dr} \qquad \forall\, s,dr, \qquad (2.1)$$

$$\sum_{s\in L_k} h_{k,s}\, Y_{s,dr} = \sum_{c}\sum_{n} X_{c,k,dr,n} \qquad \forall\, k,dr, \qquad (2.2)$$

$$X_{c,k,dr,n} \leq Z_{c,k,n}\, m \qquad \forall\, c,k,dr,n, \quad (2.3)$$

$$\sum_{k}\sum_{dr} X_{c,k,dr,n} + \sum_{k} Z_{c,k,n}\, u_k \leq G_{c,n} \qquad \forall\, c,n, \quad (2.4)$$

$$\sum_{c}\sum_{k} j_k\, X_{c,k,dr,n} = \sum_{r \in R_{dr,n}} W_{dr,n,r} \qquad \forall\, dr,n, \quad (2.5)$$

$$\sum_{dr,n}\sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \leq q_a \qquad \forall\, a, \quad (2.6)$$

$$P = \sum_{c}\sum_{n} F_{c,n}(G_{c,n}, \sum_{k}\sum_{dr} X_{c,k,dr,n}, \sum_{k} Z_{c,k,n}\, u_k, \xi) \quad (2.7)$$

$$T = \sum_{rn} H_{rn}(\sum_{dr,n}\sum_{r \in R_{dr,n}: r \in K_{rn}} W_{dr,n,r}, \xi) + \sum_{a} t_a \sum_{dr,n}\sum_{r \in R_{dr,n}: r \in J_a} W_{dr,n,r} \quad (2.8)$$

$$P + T \leq ce(\xi) \sum_{s}\sum_{dr} Y_{s,dr} \quad (2.9)$$

$$\sum_{s}\sum_{dr} R_{s,dr} \leq re(\xi)\, LD \quad (2.10)$$

$$LD \leq d(\xi) \quad (2.11)$$

$$D_{s,dr} = fr_{s,dr}(\xi)\, LD \qquad \forall\, s,dr, \quad (2.12)$$

$$Z_{c,k,n} \in \{0,1\},$$

$$D_{s,dr}, LD, R_{s,dr}, X_{c,k,dr,n}, Y_{s,dr} \geq 0.$$

Figure 8.2. Formulation of the processing capacity investment decisions model (PC1).

8.4. Modelling Stochastic Parameters

As seen in Figure 8.2, the PC1 model has stochastic parameters. When dealing with stochastic parameters it is important to differentiate between how to obtain a parameter's probability distribution and how to generate scenario instances from that distribution. This section identifies the probability distributions that underlie the

stochastic parameters, leaving the generation of scenario instances from these distributions to Chapter 11.

8.4.1. Quality of Service Customers Expect

Uncertainty is present in the parameters representing the average level of quality of service customers expect, *ce(?)* and *re(?)*, in two main ways. First, customers themselves are unlikely to know exactly what level of service they want, for example, the length of time by which they expect a service to be delivered. Each individual customer would have their own distribution, with individuals sometime reacting in different ways to the same quality of service. There is also measurement uncertainty, which comes from estimating the distribution for the population.

Because the parameters that represent the average level of quality of service customers expect are over all customers their distributions are the combination of many individual distributions. By the Central Limit Theorem Normal distributions are likely to be appropriate.

8.4.2. Maximum Demand Pool

It is likely that the maximum demand pool would be significantly more difficult to predict than the level of quality of service customers expect. This is because there is much less information available on it. Uncertainty in this parameter exists because, among other things, it is difficult to know what actually influences maximum demand, and because it could change over the capacity model's planning horizon.

We assume this parameter is represented by a Normal distribution, and because it is likely to be harder to predict than the previous parameter it should have a larger relative variance.

8.4.3. Delivery Time Functions

The delivery time functions are identified as being stochastic in Chapter 3. However, we choose not to investigate this stochasticity. This is because considering

stochasticity in delivery times adds significant complexity, and goes beyond the boundaries of this thesis. These functions are deterministic from this point on.

8.4.4. Spread of Demand ($fr_{s,dv}$)

The PC1 model determines the long-run demand for all services received from all locations. Fractions are used to then evaluate the long-run demand for a particular service received from a particular location. There is uncertainty present in these fractions, reflecting two aspects of this parameter: uncertainty and time. There is uncertainty about how demand evolves, but it is also important to consider the actual evolution of demand: what percentage of time demand is at each level.

Due to rapid changes in technology and customers' volatile demand patterns, as well as the long planning horizon for this type of problem, a service that is popular now may be obsolete by the end of the planning horizon, and vice versa, in that services with little demand now (where this includes currently non-existent services) could be popular later in the planning horizon. Similarly, locations that currently request a significant proportion of the service provider's demand might be matched or overtaken by developing markets. Also, demand for certain service / demand location combinations could spike over the planning horizon. For example, an extensive marketing campaign in a certain area for a certain service could lead to a drastic increase in demand in that area for that service. This aspects are will have a combined effect on this parameter.

8.4.5. Correlations

It is possible that the three stochastic parameters are correlated. For example, if customers know the service is very popular (high maximum demand pool) they might lower the level of quality of service with which they expect to have their service met. For the testing in Chapter 11 we assume the parameters are independent.

8.5. Summary

The two-stage stochastic model outlined in this chapter sets the processing capacity investment in the first stage under second stage uncertainty. The second stage decisions determine how to meet demand given the processing resources provided in the first stage and with the realisation of the uncertain parameters. There are trade-offs between profit potential and investment costs because providing more capacity will result in more demand, but costs more to provide. A simple outline of the two-stage model is shown in Figure 8.3. The following chapter aids understanding of this model's solution process, and Chapter 10 investigates potential model extensions.

1st stage:

Max Expected long-run profit: processing decisions profits less the costs of providing the processing resources.

s.t. Budget restrictions.

2nd stage (with uncertain parameters):

Max Processing decisions profits.

s.t. Arc and processing node capacity constraints,

Subservice installation requirements,

Meeting or rejecting demand,

Meeting level of quality of service customers expect.

Figure 8.3. PC1 model outline.

9. MODEL EXAMPLE

This short chapter aims to aid understanding about the solutions for the processing capacity expansion model. This is done by investigating potential solutions for a simple example.

9.1. Mathematical Example

By showing potential model solutions for a particular problem it is possible to gain a better understanding of the solution process for the processing capacity expansion model. In order to better achieve this understanding the mathematical model presented in Chapter 8 is substantially simplified.

The network for this example, and many of the parameter values, are shown in Figure 9.1. Only one service (composed of a single subservice) is available, which can be demanded from either node. Each location receives half of the total grouped demand ($fr_{s,dr} = 0.5$). One unit of subservice requires one unit of processing capacity, and uses one unit of arc capacity. Installing the subservice does not use any processing capacity ($u_k = 0$). Using processing or transportation capacity does not

incur any cost ($b_{c,n} = 0$, $v_a = 0$, $w_{rn} = 0$). For the piece-wise total processing time function, the slope (e_j) represents the time per unit to process at a node. For the investment cost functions, i_f and i_v represent the fixed and variable costs of investment respectively. Capacity installed incurs no maintenance ($g_{c,n} = 0$). In addition, price is $p_{s,dr} = 40$, maximum demand pool is $d = 1000$, there are 50 periods in the planning horizon, the discount rate is zero, and average level of quality of service customers expect are $ce = 6$ for delivery times and $re = 0$ for rejections. Finally, budgetary and arc constraints are not restricting.

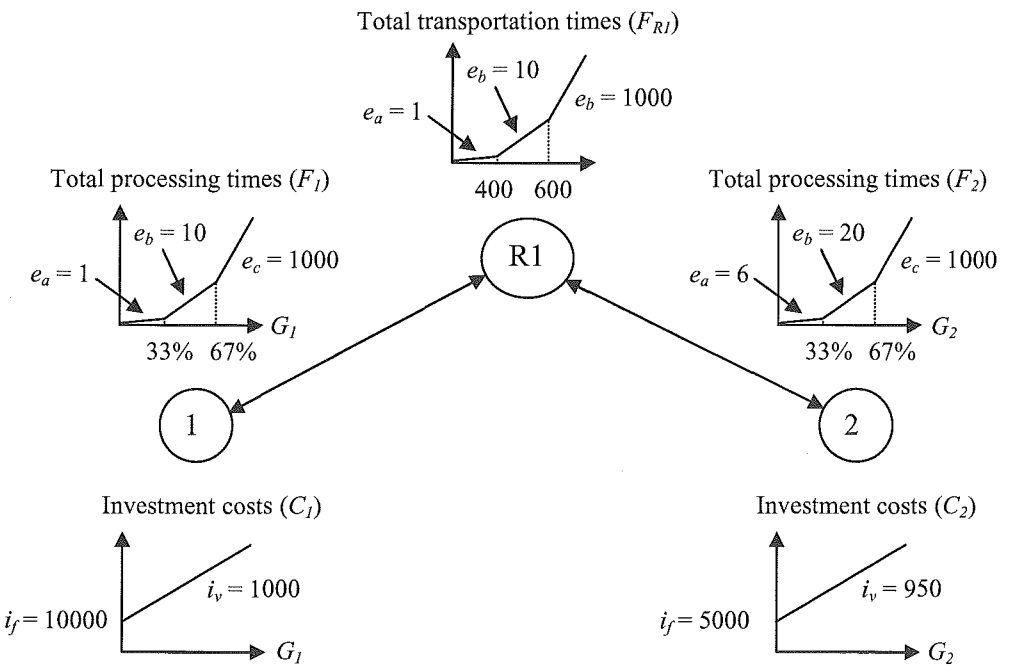


Figure 9.1. Simplified network for demonstrating PC1 model solution process.

The resulting formulation of the PC1 model for this simple example is shown in Figure 9.2 (note both stages are combined in this simple example). Potential model solutions for this problem are shown in Table 9.1.

Let:

G_n = units of processing capacity added to processing location n.

LD = total grouped long-run demand pool.

P = sum total of all processing times.

T = sum total of all transportation times.

$X_{dr,n}$ = demand received at node dr that is met at node n.

Model:

$$\textit{Maximise} \quad 50\,(40\,X_{11} + 40\,X_{12} + 40\,X_{21} + 40\,X_{22}) - C_1(G_1) - C_2(G_2)$$

(maximise long-run profit)

s.t.

$$LD \leq 1000$$

$$X_{11} + X_{12} = 0.5\,LD$$

$$X_{21} + X_{22} = 0.5\,LD \qquad \text{(meet nodal demands)}$$

$$X_{1n} + X_{2n} \leq G_n \qquad n = 1,2 \quad \text{(processing capacity restrictions)}$$

$$P = F_1(G_1, X_{11} + X_{21}) + F_2(G_2, X_{12} + X_{22})$$

$$T = F_{R1}(X_{12} + X_{21}) \qquad \text{(delivery time calculations)}$$

$$P + T \leq 6\,(X_{11} + X_{12} + X_{21} + X_{22})$$

(meeting average delivery times customers expect)

Figure 9.2. Formulation for demonstrating PC1 model solution process.

Potential PC1 Model Solutions		Expected long-run profit
Solution 1:	$G_1 = 750$, $G_2 = 1500$, $LD = 1000$, $X_{11} = 500, X_{12} = 0, X_{21} = 0, X_{22} = 500$.	–$190 000
Solution 2:	$G_1 = 1350$, $G_2 = 300$, $LD = 1000$, $X_{11} = 500, X_{12} = 0, X_{21} = 400, X_{22} = 100$.	$350 000
Solution 3:	$G_1 = 1200$, $G_2 = 0$, $LD = 800$, $X_{11} = 400, X_{12} = 0, X_{21} = 400, X_{22} = 0$.	$390 000 **OPTIMAL**

Table 9.1. Possible feasible solutions for the simplified PC1 model example.

In order to explain the solutions presented in Table 9.1 it is first important to recognise that rejections cannot be allowed for in the solutions. Solutions 1 and 2 install sufficient processing capacity to enable the maximum demand to be obtained. Solution 1 arranges this capacity so all demand is met where it was received, whereas Solution 2 arranges the capacity so that as much demand as possible is met at the faster processing node 1. Solution 3 differs, in that by requiring all demand received to be met at processing node 1, the level of long-run demand is restricted by the routing node congestion.

For this simple example, it could appear that because they enable more demand to be met, the first two capacity solutions would obtain higher long-run profits. However, as can be seen from the final column of Table 9.1, Solution 3 is in fact the best solution (and is indeed optimal). The other solutions obtain less profit because the increase in demand (and revenue) gained from the extra capacity installed does not recoup the significant investment cost in the slower processing node required to obtain this additional demand.

9.2. Stochastic Solution Process

The previous section highlighted, in a deterministic setting, the solution process for the PC1 model. This section looks at how stochasticity can impact solutions. Using the following maximum demand distribution, we show that the stochastic solution is different from the expected value solution.

Maximum Demand Distribution

To simplify the stochastic solution process the maximum demand (d) distribution is represented by two scenarios (where d equals 600 or 1000), where the scenarios have probability p_{600} and p_{1000}, respectively.

Expected Value Solution

When $p_{600} = 0.5$ and $p_{1000} = 0.5$, the expected value problem has $d = 800$, its solution is:

$$G_1 = 1200,\ G_2 = 0,\ LD = 800,$$
$$X_{11} = 400,\ X_{12} = 0,\ X_{21} = 400,\ X_{22} = 0.$$

Stochastic Solution

The wait-and-see scenario solutions for the maximum demand distribution are presented below.

Scenario 1: $d = 600$ $(p_{600} = 0.5)$.

Solution: $G_1 = 900,\ G_2 = 0,\ LD = 600,$
$X_{11} = 300,\ X_{12} = 0,\ X_{21} = 300,\ X_{22} = 0.$

Scenario 2: $d = 1000$ $(p_{1000} = 0.5)$.

Solution: $G_1 = 1200,\ G_2 = 0,\ LD = 800,$
$X_{11} = 400,\ X_{12} = 0,\ X_{21} = 400,\ X_{22} = 0.$

The expected profits of these two solutions are:

$G_1 = 900$, $G_2 = 0$:

If $d = 600$, profit = \$290 000,

If $d = 1000$, profit = \$290 000,

Hence, expected profit = \$290 000.

For $G_1 = 1200$, $G_2 = 0$:

If $d = 600$, profit = –\$10 000,

If $d = 1000$, profit = \$390 000,

Hence, expected profit = \$190 000.

In this simple example the stochastic solution is $G_1 = 900$, $G_2 = 0$, that found from the smaller of the two maximum demand scenarios, while the expected value solution is that found from the larger of the two scenarios. Note that the stochastic solution will not always be that from the smaller maximum demand scenario. For example, it can be easily seen that if the probabilities were different, such as if $p_{600} = 0.2$ and $p_{1000} = 0.8$, the stochastic solution would that of the larger maximum demand scenario. In more complicated problems the stochastic solution may not correspond to one of the wait-and-see scenario solutions.

10. EXTENSIONS

In Chapter 8 a mathematical model for the processing capacity investment decisions was developed. This chapter outlines potential extensions to this model, some of which point to future research.

10.1. Processing on Customers' Computer

Recent advances in technology, making home computers substantially more powerful, mean it is possible for services to be developed so that, instead of having to be processed by the service provider, the customer can download the program, or applet, that processes the service, and run it on their own computer. Developing a service to allow its requests to be met at both network processing nodes and the customer's computer costs extra. This section outlines the extensions to the model necessary to consider developing services to allow customers to process their own service. This extension was not considered in Chapter 8 as it adds unnecessary complexity to understanding the model.

It may not be feasible to allow the customer to process all services. For example, the service provider would have to route the entire database, causing great congestion, to enable customers to process a search engine request on their own computer.

10.1.1. Mathematical Model

The first stage decision becomes whether to develop services to use network processing ability and / or customer processing. Service development decisions are likely to be made on a different time frame from capacity expansions decisions. For the illustrative model it is assumed that the time frame for service development is known at the beginning of the planning horizon. Indices, sets, parameters, and decision variables are as defined in Chapter 8. Altered and additional parameters and decision variables are defined in Table 10.1.

Parameters

$invcc_s$ = the cost of developing service s to allow customers to process their demand on their own computers.

$h_{k,s,w}$ = the processing units of subservice k used by a request for service s, where the subservice is being met at w (w = 1 if demand is being met at a processing node, and w = 2 if demand is being met at the customer's node).

$j_{k,w}$ = the number of units of network flow required to route one unit of subservice k, where the subservice is being met at w (w = 1 if demand is being met at a processing node, and w = 2 if demand is being met at the customer's node).

$outc_k$ = the cost of transporting the flow for subservice k outside the network to the customer's computer.

$outt$ = the average time it takes to transport a unit of flow outside the network to the customer's computer.

Decision variables

CC_s = 1, if the service provider has the ability to allow customers to process demand for service s on their own computer, 0 otherwise.

$W_{dr,n,r}$ = flow from location dr being met at processing node n routed over route r between those two locations.

$W^{cc}_{dr,n,r}$ = flow between location dr and processing node n routed over route r between those two locations, when demand is being processed at the customer's computer.

$X_{c,k,dr,n}$ = processing units used by subservice k for meeting demand originating from location dr which is processed using computer c at location n.

$X^{cc}_{c,k,dr,n}$ = processing units used by subservice k for meeting demand originating from location dr which is processed at the customer's computer, but the small network processing requirement is processed using computer c at location n.

$Y_{s,dr}$ = units of demand for service s received at location dr that are met using the network's processing nodes.

$Y^{cc}_{s,dr}$ = units of demand for service s received at location dr that are met using the customer's computer.

Table 10.1. Additional parameters and decision variables when considering processing on the customer's computer.

While considering processing on the customer's computer extends the PC1 model, the model still retains the same structure. Hence, instead of repeating the model, the necessary changes are shown below.

- Because demand can now be met at processing nodes and on the customer's computer, $Y_{s,dr}$ is replaced by $Y_{s,dr} + Y^{cc}_{s,dr}$, and $X_{c,k,dr,n}$ with $X_{c,k,dr,n} + X^{cc}_{c,k,dr,n}$.

- The costs of routing outside the network ($\sum_{c}\sum_{k}\sum_{dr}\sum_{n} outc_k X^{cc}{}_{c,k,dr,n}$) must be added to the second stage objective function, and the costs of developing services allowing for processing on the customer's computer ($\sum_{s} invcc_s \; CC_s$) are added to the first stage objective function and Budget Constraint (1.1).

- Constraint set (2.2), which defines the total processing requirement for meeting demand, is replaced with:

$$\sum_{s\in L_k} h_{k,s,1}\, Y_{s,dr} = \sum_{c}\sum_{n} X_{c,k,dr,n} \qquad \forall\, k,dr, \qquad (2.2)$$

$$\sum_{s\in L_k} h_{k,s,2}\, Y^{cc}{}_{s,dr} = \sum_{c}\sum_{n} X^{cc}{}_{c,k,dr,n} \qquad \forall\, k,dr, \qquad (2.2a)$$

- Constraint set (2.3a) must be added. This constraint ensures that customers can only process their own demand if the service has been developed to allow this.

$$Y^{cc}{}_{s,dr} \le CC_s\, m \qquad \forall\, s,dr, \qquad (2.3a)$$

- Constraint set (2.5a) is added in addition to the existing constraints. This constraint defines the total network flow for demand that is intended to be processed at customer's computers.

$$\sum_{c}\sum_{k} j_k\, X^{cc}{}_{c,k,dr,n} = \sum_{r\in R_{dr,n}} W^{cc}{}_{dr,n,r} \qquad \forall\, dr,n, \qquad (2.5a)$$

- Constraint set (2.6a), which ensures the total flow over an arc does not exceed that arc's capacity, regardless of whether the total flow results from network processing or the customer processing their own request, replaces the existing constraint.

$$\sum_{dr,n}\sum_{r\in R_{dr,n}\,:\,r\in J_a} (W_{dr,n,r} + W^{cc}{}_{dr,n,r}) \le q_a \qquad \forall\, a, \qquad (2.6a)$$

- Constraint sets (2.7a) – (2.9a) replace the existing Constraint sets (2.7) – (2.9). Constraint sets (2.7a) and (2.8a) define the total time spent processing customer requests and the total time spent transporting flow respectively. The

total processing time now also includes the time spent processing on the customers' computers, which is influenced by the average processing speed of the customers' computers (F_{cc}) and the amount required to process there. The total transportation time now includes flow both inside and outside the network. Finally, Constraint set (2.9a) ensures that the average quality of service provided to all customers, with respect to delivery times, meets the average level of delivery times customers expect.

$$P = \sum_{c}\sum_{n} F_{c,n}(G_{c,n}, \sum_{k}\sum_{dr} X_{c,k,dr,n}, \sum_{k} Z_{c,k,n}\, u_k, \xi) + \sum_{s}\sum_{dr} F_{cc}(Y^{cc}{}_{s,dr}, \xi) \quad (2.7a)$$

$$T = \sum_{rn} H_{rn}(\sum_{dr,n} \sum_{r \in R_{dr,n}: r \in K_{rn}} (W_{dr,n,r} + W^{cc}{}_{dr,n,r}), \xi) + \sum_{a} t_a \sum_{dr,n} \sum_{r \in R_{dr,n}: r \in J_a} (W_{dr,n,r} + W^{cc}{}_{dr,n,r}) + outt \sum_{dr,n,r} W^{cc}{}_{dr,n,r} \quad (2.8a)$$

$$P + T \leq ce(\xi) \sum_{s}\sum_{dr} (Y_{s,dr} + Y^{cc}{}_{s,dr}) \quad (2.9a)$$

Some other important points regarding this extended model are:

- The investment costs now include the fixed cost of developing the service to be met at network processing nodes. This was unnecessary in the PC1 model, because the cost was constant regardless of how demand was met.
- When using a processing node to meet demand (w = 1) the processing is performed in the network and the routing consists of the initial request and the final results. When using a customer's computer (w = 2) the network requirements change dramatically. The service provider must route to the customer the program containing the functionality required to process the service. This is likely to entail more flow than simply routing requests and results ($j_{k,2} > j_{k,1}$). Secondly, only a small processing requirement is incurred in the network when customers process their own service; that is, to find the appropriate subservices and direct them to the customer's computer ($h_{k,s,1} > h_{k,s,2}$).

- Since each individual customer processes their own service, processing demand on the customer's computer is done on many, many computers. This is in contrast to the service provider processing all the requests using their few processing nodes, which would lead to congestion.

By making these model changes the decision as to whether to process on the customer's computer predominantly depends on the speed of that customer's computer. Information on average customer computer speed could be found from 'cookies' or by survey, for example. If a customer has a slow computer, in order to meet their expectations, the service provider should process demand in the network using its processing nodes and route the customer their results. This would be significantly faster than having the customer process requests on their slow machine. Conversely, if a customers has a fast computer, they will prefer to process their requests themselves. This is so their processing time is not affected by the congestion caused by others.

Hence, as home computer speed varies (such as over time), so will the decision as to whether to use the customer's computer to process requests. At times it may be most profitable to meet demand using processing nodes for some customers and sending the program to the customer's computer for others. This is particularly so when considering individual customers, as their computer speed will differ from the average. This is an interesting point, because in general service providers tend to only offer one method of providing the service. They either meet all demand themselves, or they always send the appropriate program(s) to the customers and get them to process their own demand.

10.2. Capacity Expansion Over Time

Thus far in this thesis capacity has been added only at the beginning of the planning horizon. The timing of the decision, or when to add capacity, has not been considered. However, there are some advantages to the service provider of delaying capacity expansion. These are:

- Stochastic parameters will be partially revealed, enabling later decisions to be made in a more informed environment.
- Delay allows the service provider to take advantage of newer technologies developed later in the planning horizon.
- Delaying expansion avoids unnecessary maintenance costs being incurred.

Including the timing aspect in the model allows the service provider to weigh these advantages against the disadvantages of delaying capacity expansion when making their decisions. These disadvantages include:

- Reduced economies of scale because smaller, more frequent, expansions do not earn the same bulk purchasing discounts.
- A greater likelihood of poor quality of service. This is because smaller expansions will be planned to allow capacity to stay just ahead of demand. If demand grows more than expected, the quality of service provided to that increased level of demand may not meet the level of quality of service customers expect, meaning demand would be lost.

Whilst adding more complexity, modelling the timing aspect of the decisions is a straightforward extension to the PC1 model. As shown in Luss (1982), the extended model would be multi-staged, where each stage represents a potential expansion point in the horizon. Importantly, when considering capacity expansions over time, high discount factors tend to delay investments until they are needed. Further investigation into this model should also consider the real options literature.

10.3. Different Classes of Customers

The models in this thesis aggregate customers, comparing average quality of service with average levels customers expect. In reality, different customers will pay different prices to receive different levels of service. To ensure that the premium customers receive better service, the model would require different groups of customers to be represented. Modelling this introduces a number of complexities:

- Customers could move between classes, where this would depend on, among other things, the quality of service they receive.
- Rules would need to be developed for deciding, when necessary, what customers were rejected. Rejecting premium customers jeopardises these most profitable customers, whilst always rejecting discount customers reduces the chances of these customers moving to more premium categories.
- The model could consider setting delivery time guarantees, where the service provider obtains more revenue for faster guarantees, but must invest in more capacity to be able to meet these promised times. As in Chapter 3, there would be implicit 'costs' associated with not meeting guarantees.

However, the most complex modelling aspect would be determining the average quality of service received by each class. This would require non-linear calculations in the PC1 model. Hence, future research could model this non-linearity in the PC1 model, or investigate alternative ways of modelling this problem situation to include this linearly.

11. COMPUTATIONAL RESULTS AND ANALYSIS

This chapter explores aspects of the PC1 model that contribute to its model complexity, analysing the impact of making model approximations to reduce this complexity. This enables a better understanding of the model.

11.1. Introduction

The processing capacity expansion decisions were modelled in Chapter 8 using a two-stage stochastic integer model (PC1). Due to a number of aspects inherent in the problem situation (stochastic parameters and integer variables in particular) this PC1 model is significantly complex, taking a prohibitive length of time to solve for realistically sized problems. This chapter investigates possible approximations to model aspects, analysing what impact making these approximations has on solutions; thereby enabling better understanding of the model and the interactions between various model aspects. By doing this we illustrate, using a generated data set, the

relative importance of the model parts. This idea is nicely summarised by Simon and Ando (1961):

> *"Justifications for approximation must be related to the decisions that depend on the approximating – if the decisions based on the approximate model are not much 'worse' than the decisions based on the more elaborate model according to some criteria, then we may be justified in using the approximate, simpler model. This consideration is strengthened if, while the improvement of the final decision is very slight, the cost of working with a larger model [such as increased solution times] is very much greater than that of working with an approximate, simpler model"*

Focusing on the aspects that particularly cause the model complexity there are six potential approximations to the PC1 model investigated in this chapter. These are:

- Approximating three stochastic parameters by including them deterministically.
- Approximating the amount of processing capacity required to install subservices at a processing node (eliminating the need for the second stage subservice installation binary variables, $Z_{c,k,n}$).
- Approximating a node's processing time function linearly (eliminating the remaining second stage integer variables).
- Approximating the integer aspect of the first stage decision variables.

It is important to note at this stage that the testing in this chapter, regarding the degradation in solution quality caused by these approximations, is done to simply provide guidelines for the appropriateness of the approximations. In no way is this testing supposed to represent a 'proof' or guarantee of quality of this approximation in all instances. Having said this, the same data sets were used when testing each potential approximation. Hence, the quality of the approximations is at least relative in this setting, suggesting which approximations appear to be better choices than others, and hence implying the relative importance of the model aspects.

11.2. Model Complexity

Aspects of the PC1 model – uncertainty, integrality, and network size (the number of processing nodes and services) – give this model significant complexity, and mean it has prohibitive solution times. Hence, when testing a potential approximation the *base* model (the model against which the *approximated* model is tested) will be an approximated version of the PC1 model. For example, for all testing the network size is approximated: five potential processing locations in the network, five subservices, and four services. Solution times for larger networks are detailed in Section 11.7. These approximations for the base model were made to ensure reasonable solution times for testing purposes.

Binary variables are used for three purposes in the model. They are used to reflect the fixed cost of investment (adding $2n$ binary variables, where n is the number of processing nodes), to account for subservice installation ($k.n.sc$, where k is the number of subservices and sc is the number of scenarios), and to ensure our linear approximation of the total processing times is appropriately represented ($2n.sc.pw(n)$, where $pw(n)$ is the number of piece-wise linear segments representing processing node n's processing time function). The total number of binary variables in the model is $2n + k.n.sc + 2n.sc.pw(n)$. In each section the number of binary variables present in the base and approximated models is noted.

The approximations are investigated in the aforementioned order because it was felt, and indeed this turned out to be correct, that stochasticity added to solution times greatly, and hence determining the impact, or lack of, of the stochastic parameters could help reduce solution times for the remaining testing.

11.3. Measuring Quality of Approximations

In order to judge the appropriateness of the various approximations, and hence the importance of the model aspect approximated, a method is needed to judge the quality of the solutions produced. The measure of solution quality used is the expected long-run profit produced by the demand simulation model, presented in Chapter 7, for the

given capacity solutions. The comparison of solution quality uses the following process:

- For a randomly chosen data set find the first stage solutions of the base and approximated models. The reader is referred to Appendix B for an outline of the data used for testing purposes and how it was generated for the model runs. All models were formulated in AMPL (Fourer *et al* (1993)), as shown in Appendix C. Because systematically allowing for rejections in a strategic environment is inadvisable, it was assumed for all testing that customers have zero tolerance towards rejections ($re(\xi) = 0$).
- Estimate the expected long-run profits of both first stage solutions. This is done using the demand simulation model (the RP model) that, for a given processing capacity solution, plays out what would happen to demand from period to period in order to estimate expected long-run profits. This model is described in Chapter 7 and explicitly detailed, including the data used, in Appendix C. Also, when applicable, different distributions for the uncertain parameters are used when estimating expected long-run profits. This is to reflect that the decision maker will not know the exact distribution when decision-making; the actual distribution is likely to be different.
- Compare the difference in expected long-run profits of the base and approximated model's first stage solutions.
- This process is repeated for many different data sets (> 100), leading to the results and conclusions presented in the following sections.

This testing process has a number of unavoidable limitations. First, the demand simulation model is a representation of how demand could play out over time; it by no means exactly portrays future demand. Second, it would be ideal for each approximation to be tested against the PC1 model (with no approximations). This cannot be done due to the prohibitive solution times for this model (as shown in Section 11.7). Also, as discussed in Section 11.2, all our testing is done for a small network size to keep solution times reasonable. Possible methods for solving larger network problems are discussed in Section 11.7.

11.4. Analysis of Approximations to the Model's Stochasticity

As shown in Chapter 8, there are a number of parameters in the model that are stochastic in nature. However, as discussed by Kall and Wallace (1994), just because the problem situation has parameters with stochasticity does not necessarily mean that this randomness should be introduced into the model. They go further by saying that the art of modelling is including, and describing accurately, important aspects of a problem situation, while excluding unimportant aspects. In this way, stochasticity of certain parameters may be present, but turn out to be unimportant. Hence, this section tests whether the stochasticity of the previously-identified parameters appears important enough to be included in the model, or whether the parameters can be included at their deterministic averages. The stochastic parameters are assumed to be independent and are tested one at a time.

To reduce solution times the base model includes the approximation of the binary subservice installation variables outlined in Section 11.5. All other integer variables remain. The base model in this section has 360 binary variables, and the approximated model has 60 binary variables. For each stochastic parameter, scenarios must be generated from the probability distributions identified in Chapter 8. The processes used are detailed throughout this section.

11.4.1. Stochasticity of Level of Delivery Times Customers Expect

Scenario Generation

When a one-dimensional parameter, such as the length of delivery time customers expect, is uncertain with a known distribution, scenarios are constructed for the model using the following method, as outlined by Kall and Wallace (1994). This method was chosen because it is simple and easily implemented, but most importantly, because it provides a reasonable representation of the continuous distribution and its moments with few scenarios.

Partition the 99% confidence interval of the distribution into 'n' equidistant subintervals. The mid-point of the subinterval is the scenario value, and the area of the distribution's curve in that subinterval its probability. This is shown in Figure 11.1 for a N(μ=7, σ=1) distribution with 15 scenarios.

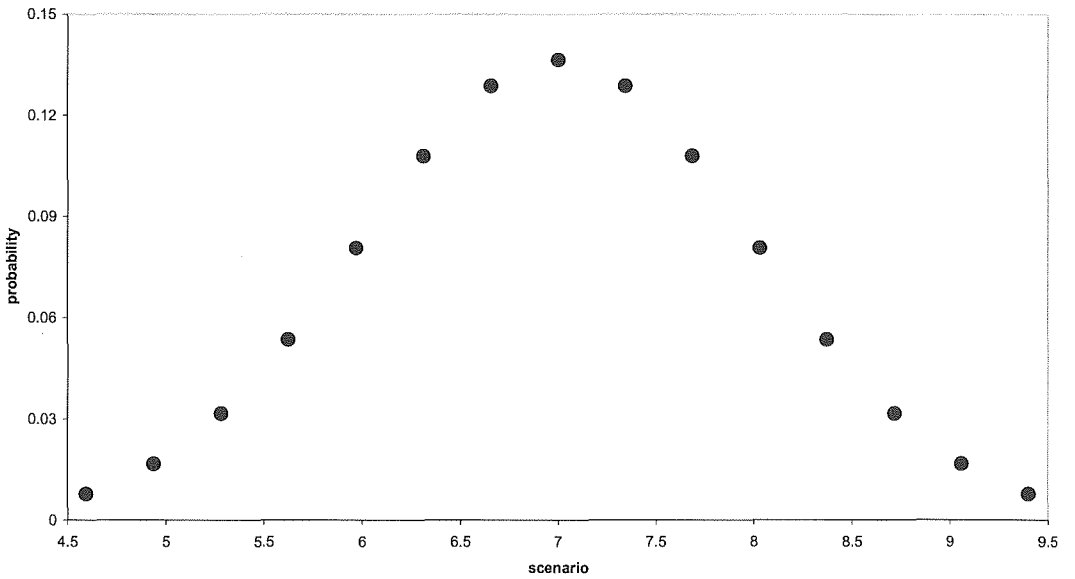


Figure 11.1. Discrete approximation to a continuous distribution, Kall and Wallace (1994).

Using this method, seven scenarios were used to discretely represent the Normal distribution for the delivery times customers expect. Specifically, a $N(7.5, 1)$[13] distribution was used. Seven scenarios were chosen because testing showed that using this number of scenarios provided a reasonable approximation of uncertainty for a small increase in solution time. However, because the first stage decisions are fixed, the expected long-run profits are estimated in the RP model under more scenarios. These n scenarios (50) are randomly sampled from another distribution for the parameter, with each scenario assigned a probability of $1/n$. This distribution is

[13] $N(7.5, 1)$ represents a Normal distribution with $\mu = 7.5$, $\sigma = 1$.

Normal, with mean and standard deviation perturbed slightly ($\mu = \mu + U(-1,1)$ and $\sigma = \sigma + U(-0.2,0.2)$), recognising that whilst it is possible to estimate the probability distribution, we are unlikely to have it exact.

Results

Using the testing procedure outlined in Section 11.3 and the scenario generation process described above, this section presents results that were obtained for the difference in solution quality between the base (stochastic) and approximated (deterministic) models.

Stochastic Parameter: Delivery Times Customers Expect	
Mean percentage difference in expected long-run profits:	0.9%
95% of observations have percentage differences between:	(-5.1%, 9.1%)
Proportion of observations with percentage differences within ±5%:	0.88
Proportion of time there is no difference between both models:	0.52

Table 11.1. Summary of results for approximation of stochasticity of delivery times customers expect.

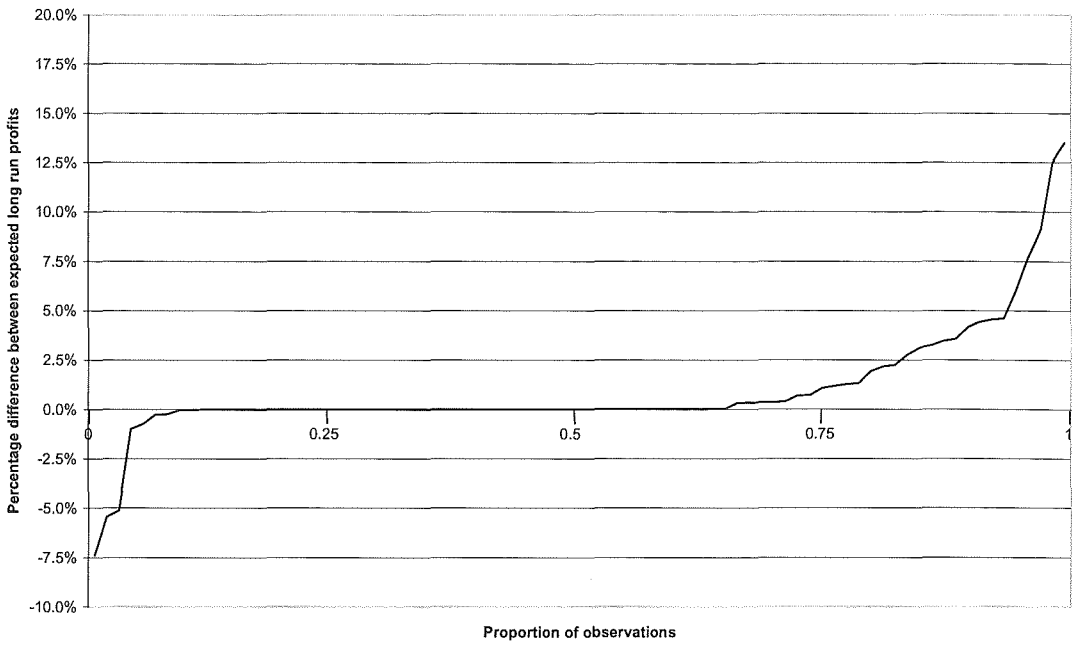


Figure 11.2. Plot of results for approximation of stochasticity of delivery times customers expect.

Table 11.1 shows that, while the expected long-run profits of the solutions produced by the approximated (deterministic) model are worse[14] than that of the base (stochastic) model's solutions, the average percentage difference is very small (0.9%). Indeed, as is also shown in Figure 11.2[15], 88% of observations have percentage differences within ±5%. The same analysis was conducted for different distributions (with larger and smaller variances) and the results were comparable.

From the results presented in this section it appears as though little is lost in terms of solution quality when making this deterministic approximation. A benefit of making

---

[14] In all sections a positive difference indicates that the base model's solution has a higher expected long-run profit.

[15] The *x* axis on this figure is the proportion of observations with a percentage difference less than this. The scale of the *y* axis is kept the same for most of the results in this chapter to easily show the relative differences between approximations.

this approximation is significantly faster solution times, as seen in Table 11.2[16]. Note that these solution times are only for the small network problem used in this testing; the difference in solution speed would be even more pronounced had larger networks been used.

Solution times	Base	Approx.
Mean solution time (seconds)	**631.5**	**1.7**
95% of observations had solution times between (seconds):	(110, 2800)	(1.0, 3.7)

Table 11.2. Solution times for approximating stochasticity of delivery times customers expect.

11.4.2. Maximum Demand Pool Stochasticity

Scenario Generation

Scenarios for the Normal distribution representing the maximum demand pool parameter were constructed using the same method as for the parameter representing the delivery times customers expect. Seven scenarios were again used, constructed from a $N(10\ 000, 1000)$ distribution. The solutions were again tested under different distributions in the RP model (where $\mu = U(0.8,1.2)\ \mu$, and $\sigma = U(0.75,1.5)\ \sigma$).

Results

Using the testing procedure outlined in Section 11.3 and the scenario generation process described above, this section presents results that were obtained for the

[16] All solution times and results were found using CPLEX 7.1 on a computer with CPU speed 1.53 GHz and 512 RAM.

difference in solution quality between the base (stochastic) and approximated (deterministic) models.

Stochastic Parameter: Maximum Demand Pool	
Mean percentage difference in expected long-run profits:	1.8%
95% of observations have percentage differences between:	(-4.4%, 6.4%)
Proportion of observations with percentage differences within ±5%:	0.82

Table 11.3. Summary of results for approximation of maximum demand pool stochasticity.

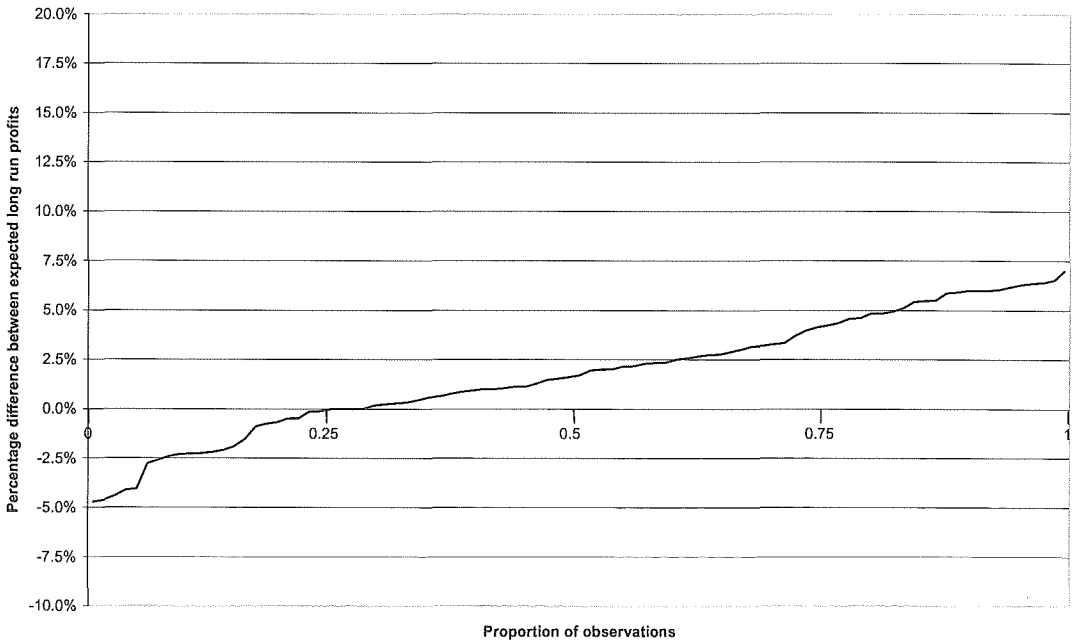


Figure 11.3. Plot of results for approximation of maximum demand pool stochasticity.

Similarly to the results in the previous section, Table 11.3 and Figure 11.3 show that, while the expected long-run profits of the solutions produced by the approximated

model are worse than that of the stochastic model's solutions, the average percentage difference is still small (1.8%). Indeed, the percentage differences rarely go beyond ±5%. From these results it appears that it is possible to approximate the PC1 model by excluding the stochastic nature of the maximum demand pool parameter. Importantly, reinforcing this conclusion, the same analysis was conducted for different maximum demand pool distributions (including distributions with greater variability) and the results were comparable. As seen in Table 11.4, the difference in solution times of the stochastic and deterministic models is similar to that in the previous section.

Solution times	**Base**	**Approx.**
Mean solution time (seconds)	**631.1**	**1.7**
95% of observations had solution times between (seconds):	(58, 4200)	(1, 3.7)

Table 11.4. Solution times for maximum demand pool stochasticity approximation.

11.4.3. Spread of Demand Stochasticity ($fr_{s,dv}$)

Scenario Generation

The stochasticity of the spread of demand parameter is more difficult than the previous two parameters because this parameter has more than one dimension. To model the uncertainty of this parameter, construction could be used. For example, Høyland and Wallace (2001) use non-linear programming to closely match the statistical measures of any continuous distribution when constructing scenarios. These techniques are used to generate stable scenarios, that is, the solution is robust with respect to the choice of scenarios. We sample scenarios, meaning solution stability is unlikely to be present. However, using the small number of scenarios as we are forced to do in this testing, construction would be unlikely to cause this

outcome anyway. Our sampling process is described below, and is explicitly shown in Appendix C.

Base Case

Regardless of events over the planning horizon, some service and demand locations are likely to always receive more demand than others. For example, a large demand location (such as a large city) is likely to always have greater demand for a particular service than a small demand location (such as a small town). Also, the size of a location is unlikely to drastically change over the planning horizon. However, whilst being true for demand locations, this reasoning is not necessarily true for services. A service that is popular now may be obsolete by the end of the planning horizon, and vice versa, in that services with little demand now (where this includes currently non-existent services) could be popular later in the planning horizon.

Because of this, the spread of demand over all service / demand location combinations is viewed as having an underlying base, where this takes into account the size of the demand location.

Scenarios

It is from this underlying base that, due to events during the planning horizon, the actual spread of demand (for a scenario) comes from. The degree to which a spread of demand scenario differs from the underlying base is randomly sampled in the following manner.

Whether a service or location's demand changes over the planning horizon, or whether it remains relatively constant, is randomly chosen for each run. The degree to which it changes is randomly sampled for each scenario in that run. Finally, demand for certain service / demand location combinations could spike over the planning horizon. These changes can happen to a different number of service / demand location combinations, to different combinations, and by different amounts in each scenario.

Once the changes from the underlying base due to the aforementioned factors have been determined, the spread of demand parameters for each scenario are re-scaled so they sum to one.

The above process gives the required number of randomly sampled spread of demand ($fr_{s,dr}$) scenarios for each run. However, because of the random nature of the sampling, and the limited number of scenarios being sampled, the underlying base is also always included as a scenario. Finally, when the spread of demand parameter is included deterministically in the model it is included at its underlying base.

Seven scenarios were still used to represent the stochasticity of this parameter. This is because, as seen in Figure 11.4, the solution times increase significantly for more scenarios, making testing impractical.

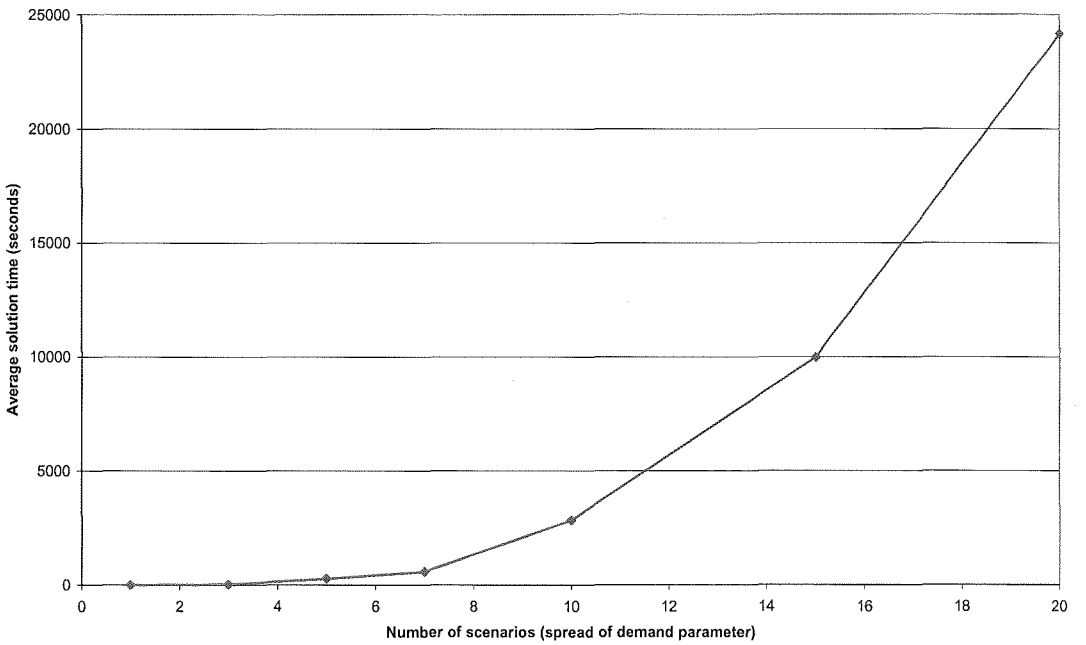


Figure 11.4. Average solution times versus number of spread of demand scenarios.

Results

Using the testing procedure outlined in Section 11.3 and the scenario generation process described above, this section presents results that were obtained for the

difference in solution quality between the base (stochastic) and approximated (deterministic) models.

Stochastic Parameter: Spread of Demand	
Mean percentage difference in expected long-run profits:	3.3%
95% of observations have percentage differences between:	(-1.9.%, 15.4%)
Proportion of observations with percentage differences within ±5%:	0.70

Table 11.5. Summary of results for approximation of spread of demand stochasticity.

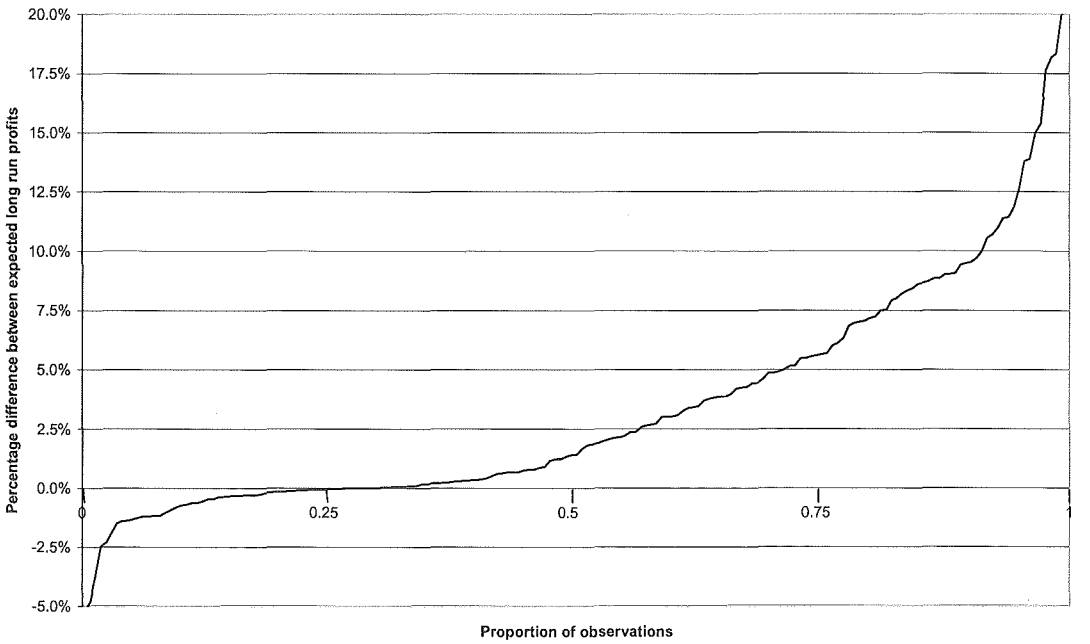


Figure 11.5. Plot of results for approximation of spread of demand stochasticity.

Unlike for the previous two parameters, where it was reasonably clear that the approximation had little impact, Table 11.5 and Figure 11.5 both suggest that in this case there is a significant difference between the expected long-run profits of the base

and approximated models' solutions. Hence, it appears that the stochasticity of this spread of demand parameter is important in the model, as the solutions are more often improved by including the stochasticity. However, the trade-off for not making this approximation is increased solution times, as shown in Table 11.6. Note that if methods were used to better consider spread of demand stochasticity, such as if more scenarios were used, or the scenarios were chosen using an appropriate construction technique, we would expect the results to be even more pronounced.

Solution times	Base	Approx.
Mean solution time (seconds)	**785.9**	**1.4**
95% of observations had solution times between (seconds):	(60, 4800)	(1.0, 3.2)

Table 11.6. Solution times for spread of demand stochasticity approximation.

11.5. Analysis of Approximating the Use of Binary Variables for Subservice Installation

Using the binary subservice installation variables ($Z_{c,k,n}$) in the model allows the amount of processing capacity used to install subservices (*PCSUBS*), and hence the associated impact on the average processing times, to be determined exactly. However, the trade-off for this level of accuracy is significantly more model complexity, than a model that, say, linearly estimates *PCSUBS*. This section firstly looks at linear approximations, and then investigates whether these approximations of *PCSUBS* appear appropriate. Note that Tomasgard (1998) removes this complexity by not considering *PCSUBS* in his strategic model.

The base model in this section, against which the *PCSUBS* linear approximation is tested, is deterministic, but has all the integer variables included. Including spread of demand stochasticity, identified as being important in Section 11.4, would make

solution times prohibitive for testing purposes. In this section the base model has 85 binary variables, and the approximated model has 60 binary variables.

11.5.1. Linear Approximation for PCSUBS

The premise behind the linear approximation for *PCSUBS* is that the amount of processing capacity used to install subservices in the network can be estimated relatively accurately without the need for binary variables in the model. There are two main methods under which this could be done. These are: relaxing the binary element of the subservice installation variables and allowing them to be continuous variables between 0 and 1 in the model, or removing this variable entirely and using model data to estimate *PCSUBS*.

Continuous 0-1 Variables

Relaxing the binary element of the subservice installation variables allows the *PCSUBS* estimation to still be made in the model environment, using model variables. However, using continuous variables no longer achieves what this variable set intended to do, namely, to ensure a fixed amount of processing capacity was sacrificed to enable demand to be met at a processing node. Hence, this approximation under-estimates the amount of processing capacity used to install subservices in reality, which leads to an under-estimation of the amount of processing capacity installed, and hence lower demand when tested in the real environment. Initial testing confirmed this approximation was inaccurate, and hence it was not considered further.

Pre-Specified Constant

Another approximation is to use an estimated constant for *PCSUBS*. This approximation can be summarised by showing the total amount of processing capacity used of computer c at processing node n under either method.

Integer model: $\sum_{k}\sum_{dr} X_{c,k,dr,n} + \sum_{k} Z_{c,k,n} u_k$

Linear model: $\sum_{k}\sum_{dr} X_{c,k,dr,n}$ + Estimated constant

Using this estimation method also eliminates the need for the PC1 model's constraint set (2.3), seen below.

$$X_{c,k,dr,n} \leq Z_{c,k,n}\, m \qquad \forall\ c,k,dr,n,\ (2.3)$$

The actual constant used will of course depend on the model instance. The next section looks at how to choose this value.

11.5.2. Estimating PCSUBS Using Model Data

This section outlines how model data could be used to estimate *PCSUBS*. The following parameters, most of which are model constants (except pc_n and pnc_n, which are the results of first stage decisions), were considered to have a significant impact on *PCSUBS*.

- The amount of processing capacity it takes to install each subservice (u_k),
- The number of different subservices (K),
- The size of the processing nodes (pc_n),
- The spread of demand ($fr_{s,dr}$),
- The processing node's characteristics (cost, processing time rates and breakpoints) (pnc_n),
- The routing node's characteristics (cost, transportation time rates and breakpoints) (rnc_{rn}).

Multiple linear regression was used to identify which independent variables were the best estimators of the *PCSUBS* dependent variable[17]. The co-efficients for the independent variables were then found by enumeration[18]. This process was done using two methods: where *PCSUBS* was constant for all computers, and where *PCSUBS* was dependent on the computer's processing capacity and characteristics. The best estimate was found to be:

$$PCSUBS_n = 2.4 \frac{\sum_k u_k}{N} \qquad (N \text{ is the number of processing nodes})$$

Note that, even though this approximation removes the subservice installation binary variables, the second stage still has integer variables. These are discussed in Section 11.6.

11.5.3. Computational Analysis of Approximation

Using the testing procedure outlined in Section 11.3, this section presents results that were obtained for the difference in solution quality between the base and approximated models. The sample used to test this approximation was independent of the one used to determine the co-efficients.

---

[17] To produce observations for what level of processing capacity was actually used to install subservices given varying values of the above independent parameters, the base model was run many times. The data for each run was chosen as discussed in Appendix B.

[18] As regression chooses the co-efficients to most accurately estimate *PCSUBS*, co-efficients were chosen by enumeration (to two significant figures) to most closely (by Least Squares) approximate the base model's expected long-run profits.

***PCSUBS* Linear Approximation**	
Mean percentage difference in expected long-run profits:	0.3%
95% of observations have percentage differences between:	(-0.9%, 2.9%)
Proportion of observations with percentage differences within ±5%:	1

Table 11.7. Summary of results for *PCSUBS* approximation.

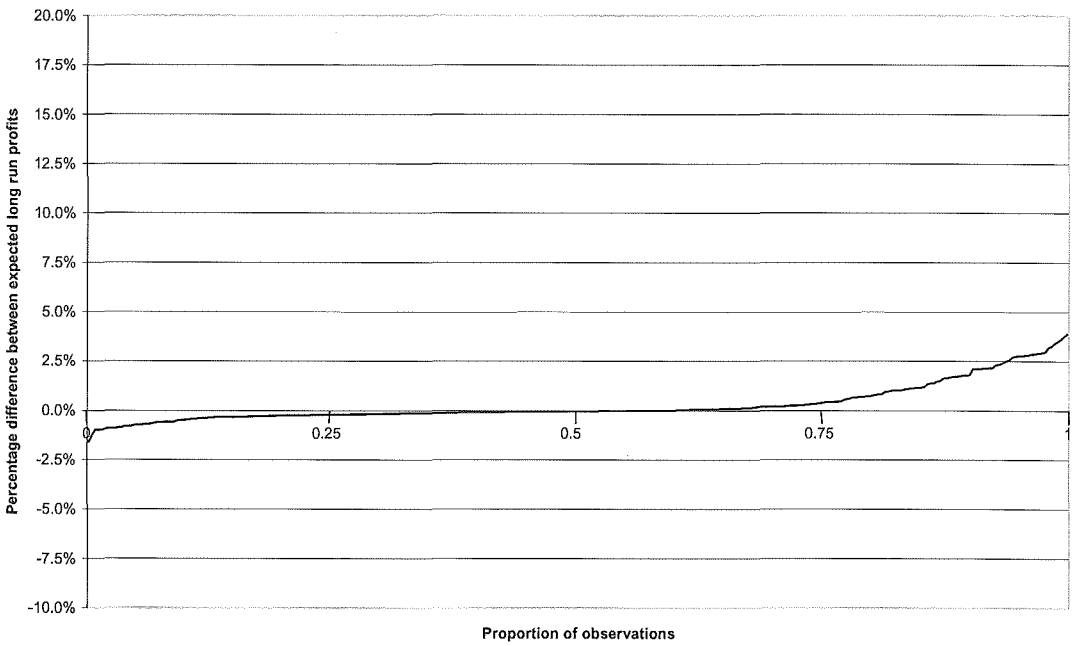


Figure 11.6. Plot of results for *PCSUBS* approximation.

Table 11.7 shows that the expected long-run profits of the solutions produced by the approximation model are very similar to those of the base model, with the mean percentage difference being very small (0.3%). The range of percentage differences between expected long-run profits is also very small, with 95% of observations between -0.9% and 2.9%, and all observations within ±5%. Information about the percentage differences for all observations can also be seen from Figure 11.6.

From the testing in this section it appears that using the linear approximation for *PCSUBS* has little impact on the quality of solutions. Because the numbers of binary variables are relatively similar, the solution times are not substantially improved by the approximation (see Table 11.8). However, the difference is likely to be extenuated when solving larger network problems.

Solution times	Base	Approx.
Mean solution time (seconds)	**3.4**	**1.5**
95% of observations had solution times between (seconds):	(1.4, 8.0)	(1.0, 3.5)

Table 11.8. Solution times for *PCSUBS* approximation.

11.6. Analysis of Approximations to the Model's Remaining Integer Variables

The model's remaining two sets of integer variables could be approximated to further simplify the model. These approximations are tested in this section, using the same method as in the previous sections. The remaining second stage integer variables were relaxed first, because there are more of them and they are less influential in determining the processing capacity solution.

The base model uses the *PCSUBS* approximation outlined in Section 11.5, but all other integer variables remain. Also, using the results from Section 11.4, the spread of demand parameter is included stochastically, whilst the other uncertain parameters are included deterministically. Hence, the base model has 360 binary variables.

11.6.1. Linear Processing Time Function

- Approximating a node's processing time function linearly (eliminating the remaining second stage integer variables) (L_{2nd} approximation).

The second stage of the PC1 model has binary variables to incorporate the convex nature of the processing time function (as shown in Figure 11.7a). The binary variables are needed to ensure that the processing times approximation, discussed in Chapter 5, is modelled properly. This approximation removes these integer variables by using a linear approximation of this processing time function (as seen in Figure 11.7b). The integer variables are no longer necessary for this approximation because there is now a constant processing time rate for all processing capacity used (and only one segment in the unit approximation function). A linear approximation was used to eliminate integrality, rather than simply making the binary variables continuous, so that good approximations of the function could be chosen.

The linear approximation chosen is similar to that investigated in Section 4.3.2 of Chapter 4[19]. For this to be used an 'effective capacity' constraint (as suggested by Le Blanc *et al* (1999)) must be added to ensure quality of service does not deteriorate beyond a pre-defined level. Le Blanc *et al* give little detail as to how to choose this proportion. Hence, in this research the proportion of processing capacity corresponding to the final breakpoint in the original piece-wise linear function (x%) was used. This approximation leaves the model with ten binary variables.

[19] By looking at what type of solutions this model approximation can produce, Chapter 4 concluded that this approximation was inappropriate. Here we confirm this conclusion in the strategic environment.

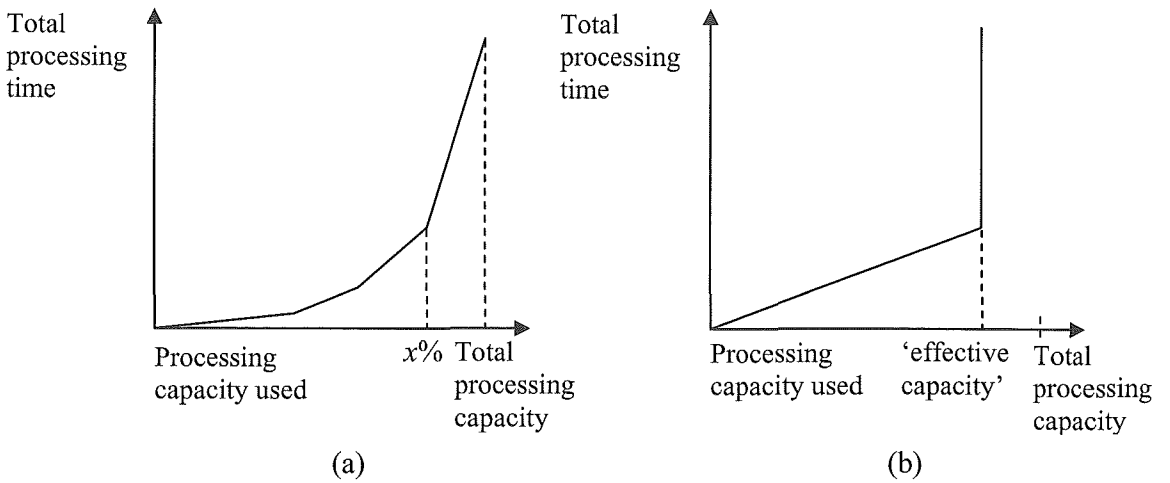


Figure 11.7. Effect on processing time function of removing the remaining second stage binary variables.

11.6.2. Linear First Stage Variables

- Approximating the integer aspect of the first stage decision variables ($L_{1st/2nd}$ approximation).

The first stage has integer variables to allow the fixed component of the investment cost function to be properly included. As well as making the above linear approximation (thereby removing all the second stage integer variables), this approximation also relaxes the integer requirement of the model's first stage decision variables by making these variables linear. The resulting model is a linear program.

11.6.3. Computational Analysis of Approximations

Using the testing procedure outlined in Section 11.3 and the scenario generation process for the stochastic spread of demand parameter outlined previously, this section presents results that were obtained for the difference in solution quality between the base and approximated models.

Integrality Approximations:	L_{2nd}	$L_{1st/2nd}$
Mean percentage difference in expected long-run profits:	2.0%	8.2%
95% of observations have percentage differences between:	(-6.7%, 17.1%)	(-5.3%, 34.4%)
Proportion of observations with percentage differences within ±5%:	0.78	0.34

Table 11.9. Summary of results for approximation of model's remaining integer variables.

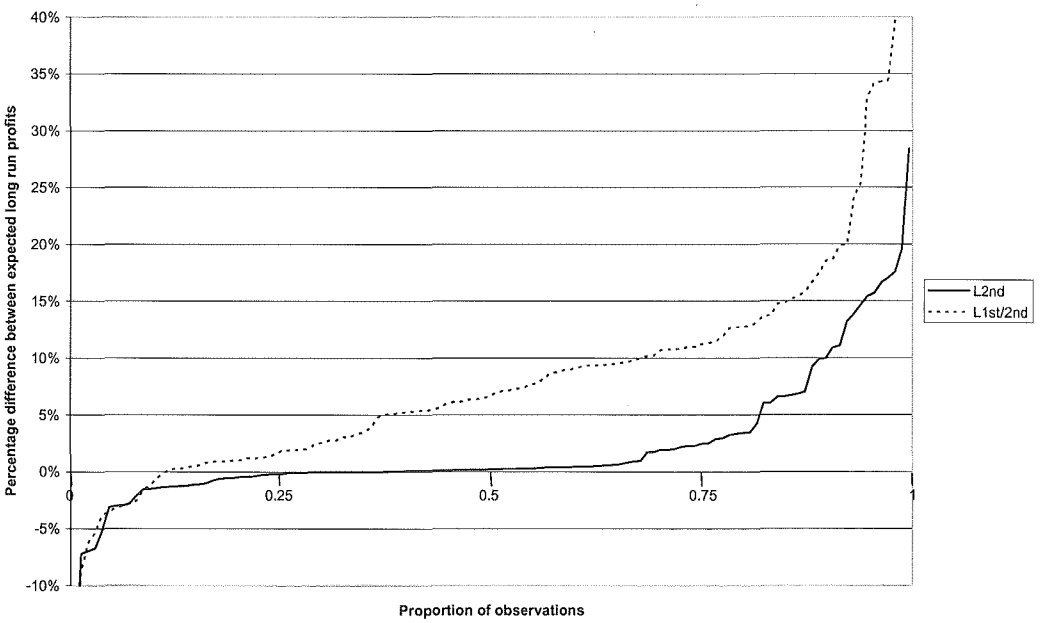


Figure 11.8. Plot of results for approximation of model's remaining integer variables.

From Table 11.9 and Figure 11.8 it appears that, as expected, approximating the first and second stage integer variables linearly ($L_{1st/2nd}$) is not a reasonable approximation to the processing capacity expansion decision model. The $L_{1st/2nd}$ approximation generally resulted in a significantly lower expected long-run profit. This is because, without the binary aspect, the fixed cost for investment is virtually eliminated, meaning the solution derived under these conditions is likely to have more processing

nodes / computers installed than is optimal, and hence produce a worse expected long-run profit when evaluated in the proper environment.

It also appears that, when using the L_{2nd} approximation, there is a significant difference between the expected long-run profits of the respective models, at least relative to the other approximations tested in this chapter. This result was also expected, because using the knowledge gained from Chapter 4, a model with this approximation (which has constant marginal costs) would not produce strictly 'distributed processing' solutions[20].

As seen in Table 11.10, both these approximations reduce solution times dramatically.

Solution times	Base	L_{1st}	$L_{1st/2nd}$
Mean solution time (seconds)	**587.4**	**22.6**	**1.2**
95% of observations had solution times between (seconds):	(99, 1800)	(3.6, 86)	(0.7, 2.3)

Table 11.10. Solution times for integrality approximations.

11.7. Solving Larger Network Problems

The aim of this section is to provide the service provider with an idea of how long it takes various base models to solve larger problems, and hence what approximations they must make to solve their problem size in reasonable time. The solution times are found for five base models:

- DeterInt – no stochasticity and all integer variables,
- StochInt – spread of demand stochasticity and all integer variables,

[20] Note that Chapter 4 defines solution types while considering the increasing marginal delivery times in the objective function. In this chapter these delivery times are in the constraints. However, the same reasoning as discussed in Chapter 4 applies, and hence the same operational solution types occur.

- StochPCSUBS – spread of demand stochasticity and *PCSUBS* approximation,
- StochL2nd – spread of demand stochasticity, *PCSUBS* approximation, and the L_{2nd} approximation,
- StochL1st/L2nd – spread of demand stochasticity, *PCSUBS* approximation, and the $L_{1st/2nd}$ approximation.

11.7.1. Problem Size

This section details model solution times as the problem size (number of processing nodes and services) increases. Solution times for the base models are found until the number of processing nodes or services makes the model's solution time exceed 10 000 seconds. For more processing nodes and services the average solution times are greater than this. Results are presented in Figure 11.9, Figure 11.10, and Table 11.11.

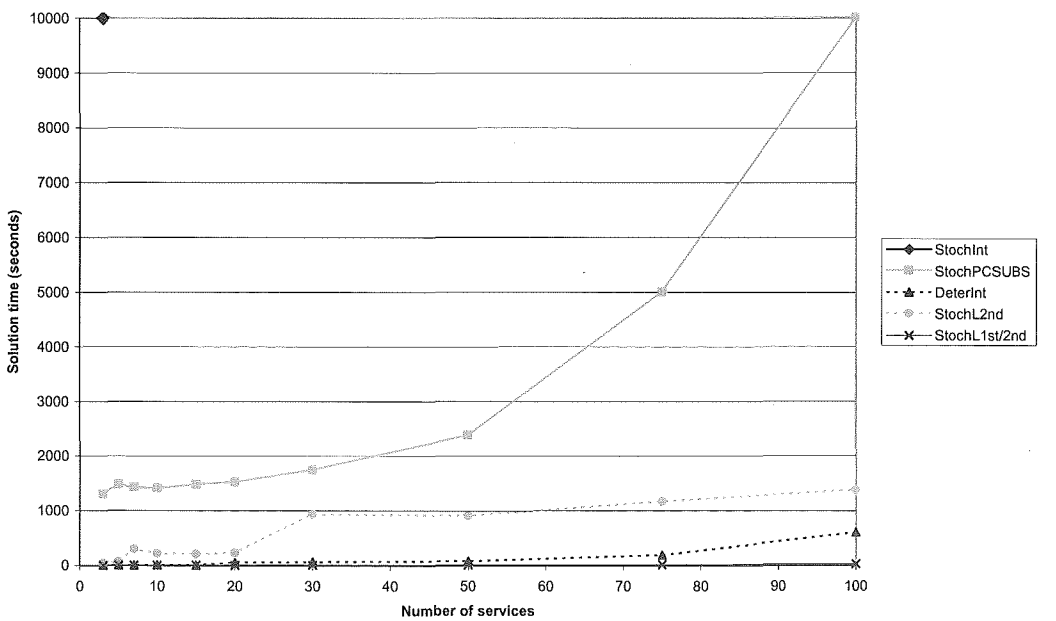


Figure 11.9. Average solution time for the various models for different number of services (five processing nodes).

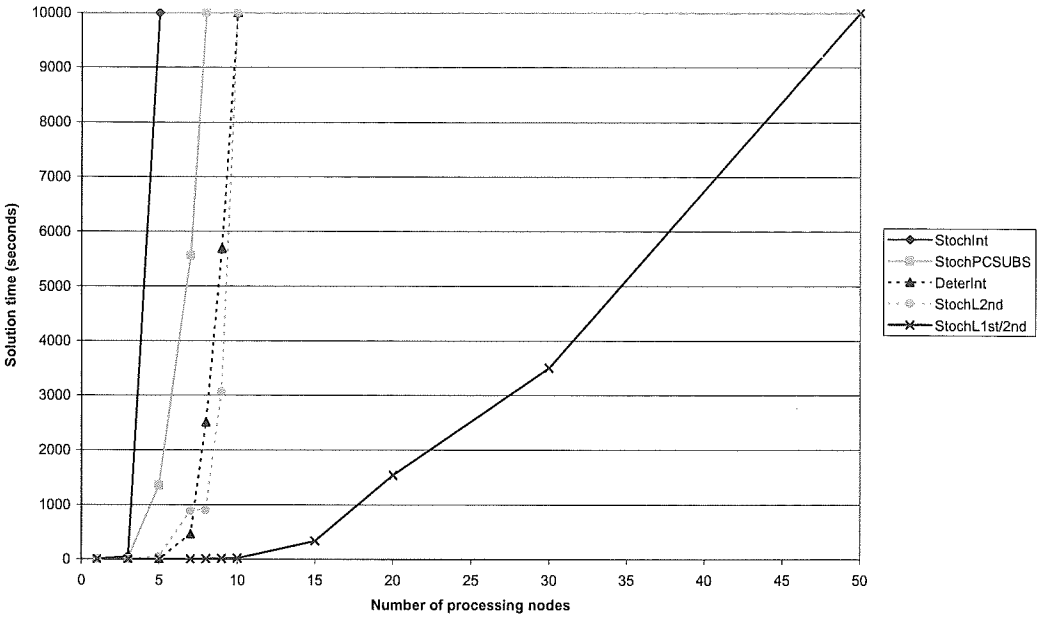


Figure 11.10. Average solution time for the various models for different number of processing nodes (four services).

Number of processing nodes (4 services)	StochInt	StochPCSUBS	DeterInt	StochL2nd	StochL1st/2nd	Number of services (5 processing nodes)	StochInt	StochPCSUBS	DeterInt	StochL2nd	StochL1st/2nd
1	0.79	0.73	0.71	0.69	0.28	**3**	10000	1306	2.16	39.17	1.40
3	54.02	6.18	0.87	0.95	0.69	**5**		1480	3.44	63.57	1.42
5	10000	1351	3.05	45.37	1.41	**7**		1424	3.3	299.8	1.13
7		5560	460.8	881.1	3.95	**10**		1408	8.8	218.4	1.26
8		10000	2512	900	8.26	**15**		1482	6	210	2.12
9			5689	3067	11.67	**20**		1528	58.3	231.2	3.08
10			10000	10000	20.58	**30**		1750	63.52	936	5
20					1540	**50**		2380	80.2	904	9.1
30					3500	**75**		5000	190.8	1170	13.1
50					10000	**100**		10000	599.4	1376	23

Table 11.11. Average solution times for the various models for different number of processing nodes and services.

The analysis of average solution times as problem size increases produces some interesting results. Average solution times increase faster as more processing nodes are added. Larger, more realistic, networks quickly take too long to solve. This confirms that the testing in this chapter could only be done on small network problems.

However, in reality, service providers are likely to have networks with many potential processing locations, and our model should allow for this. One possible method is to reduce a network with, for example, thirty potential processing locations, by grouping close locations into six groups of five. The network with six processing nodes could be solved in reasonable time, and the results extrapolated back to the thirty-node problem. Initial testing has confirmed this method has potential, but further testing, outside the bounds of this research, is necessary to confirm this, and to fine-tune the method.

11.7.2. Stochastic Scenarios

It is also important to investigate the impact of increased accuracy in approximating the stochastic distributions (number of scenarios) on model solution times. Because the computational burden is already great for most base models, even with only a small number of processing nodes, it is only worthwhile conducting this investigation for the StochL1st/L2nd and StochL2nd model approximations. Figure 11.11 shows the problem size (varying the number of processing nodes and the number of spread of demand scenarios) that could be solved in 3600 seconds for these model approximations. Because, as seen in Figure 11.9, the solution times are reasonably constant for both these models as the number of services increases, the number of services was arbitrarily fixed at ten.

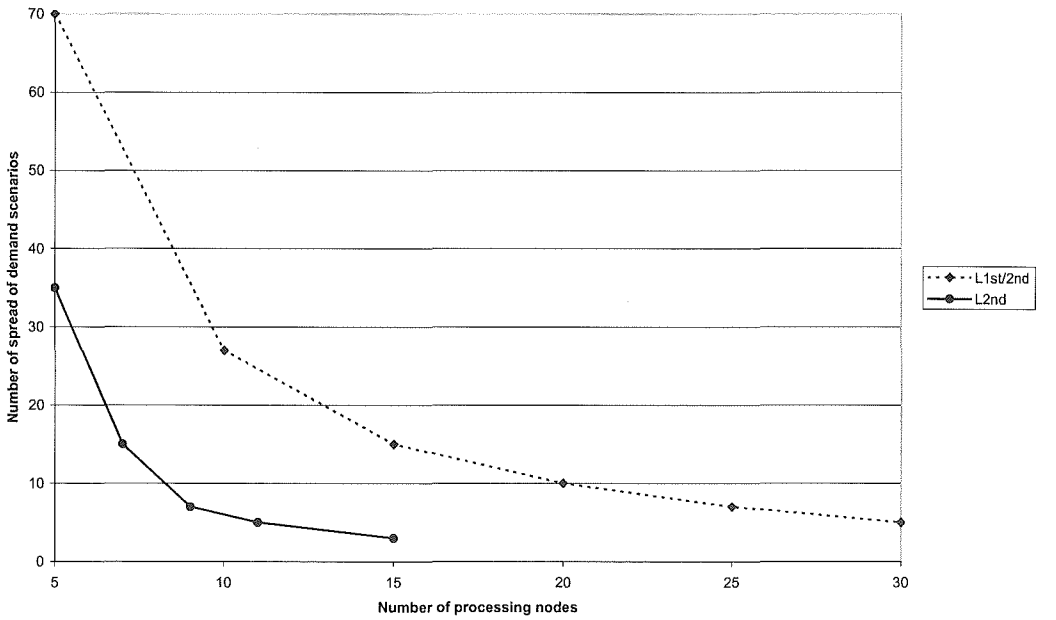


Figure 11.11. Problem size solved in 3600 seconds.

11.7.3. Processing Time Function Approximation

Finally, Figure 11.12 shows the average solution times as the number of linear segments used to approximate the processing time function increases. This is important, because the more segments in the processing time function, the more

integer variables in the model – the trade-off, of course, being the increase in accuracy with which the processing time function is represented in the model. This figure suggests that a reasonable approximation of the processing time function can be made (up to ten segments) without significantly affecting solution times.

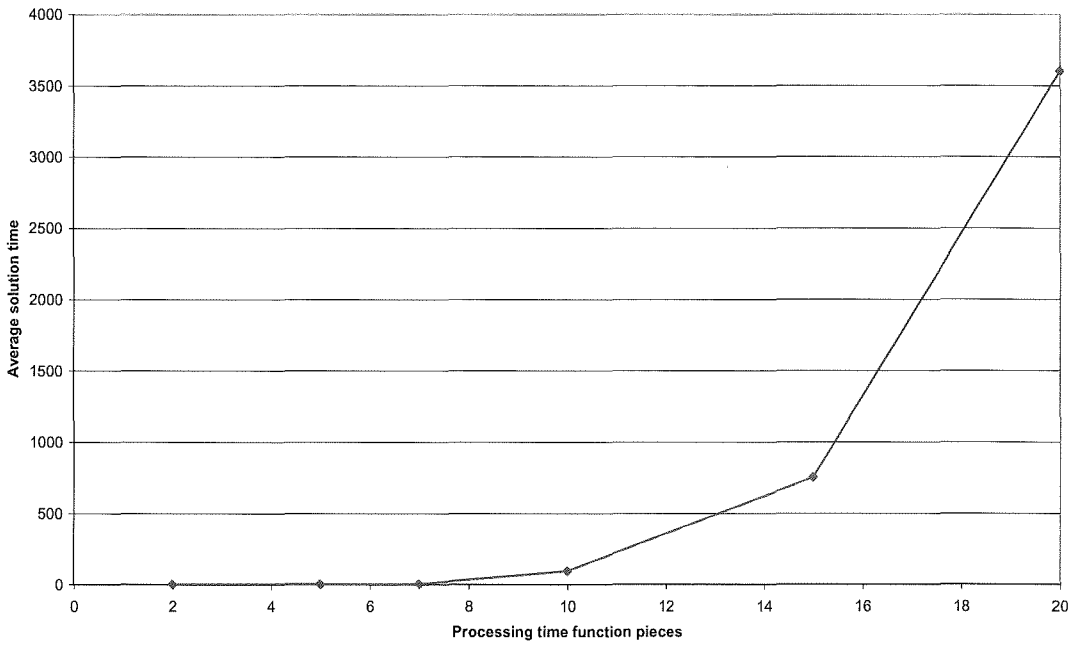


Figure 11.12. Average solution time for DeterInt model as the number of segments in the processing time function increase.

11.8. Comparing Endogenous Demand With Exogenous Demand

As discussed in Chapter 6, in this research we have modelled demand as an endogenous variable, rather than using an exogenous forecasted estimate. However, as using the demand forecast is simpler, it is important to investigate whether the extra complexity associated with modelling demand behaviour is necessary.

11.8.1. Testing Process

The testing process used for this analysis is the same as used throughout Chapter 11. The base model is the deterministic PC1 model that models demand, and has integer

variables (other than the subservice installation variables) included. The approximated model is the model that uses a forecasted estimate for demand. Hence, instead of demand being an endogenous variable in this model, it is constant with known, forecasted, demand scenarios. The demand scenarios are estimated based on what the long-run demand was for the base model. This biases the testing in favour of the forecasted demand model, in reality, their forecasted estimates for demand could be much worse than this. Effective capacity constraints were added to the approximated model to consider quality of service, in the same manner as described in Section 11.6.1.

11.8.2. Computational Analysis

Table 11.12 and Figure 11.13 show the results of the comparison between the base and approximated models' solutions.

	Expected long-run profits
Mean Percentage Difference	6.68%
95% of observations are between:	(-1.58%, 27.04%)

Table 11.12. Summary of results for endogenous versus exogenous demand.

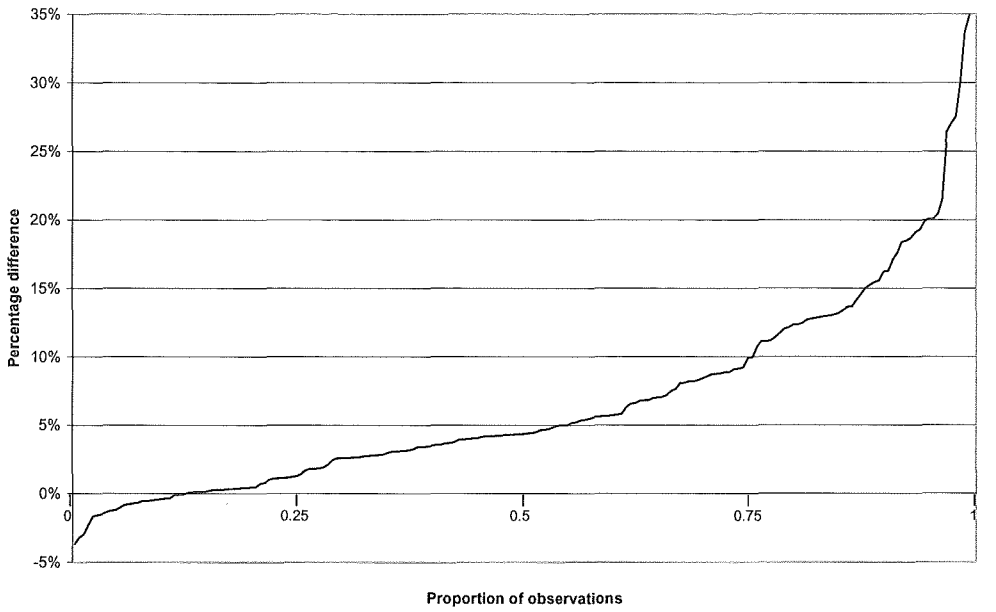


Figure 11.13. Plot of results for endogenous versus exogenous demand.

As can be seen from these results, the model with forecasted demand has significantly lower expected long-run profits (on average 6.68% worse) than the base model that models demand as an endogenous variable. This result was caused because when demand is included as an endogenous variable in the model the impact quality of service has on demand can be appropriately evaluated. This can be done more accurately when demand is a model variable, than when effective capacities are used. However, these results should be viewed with caution because they could be biased by our choice of 'effective capacity'. Intuition still suggests the implied result.

11.9. Conclusion

The results presented in this chapter offer important insight as to the impact of approximating aspects of model complexity. To test the degradation of solution quality brought about by a model approximation, solutions were found using base and approximated models, and the expected long-run profit for each model solution was estimated using a demand simulation model. The expected reduction in long-run profits, and hence solution quality, could then easily be identified. The computational

testing presented within suggested, for the generated environment, the following guidelines as to the relative importance of model aspects:

- It appears important to include the stochasticity of the spread of demand parameter in the model. Conversely, the delivery times customers expect and the maximum demand pool parameters can be included at their deterministic averages.
- The amount of processing capacity used to install subservices can be approximated with a constant amount, which is estimated using model data.
- The second stage integer variables, required to include the piece-wise nature of the processing time function, and the integer aspect of the first stage variables, should remain.

In reality, service providers will choose the model approximations which degrade solution quality the least, whilst allowing the model to be solved in reasonable time. These decisions will depend on the size of the service provider's network and the amount of solution time they have available. This is because, with a smaller network, fewer model approximations will need to be made to ensure reasonable solution times.

12. CONCLUSIONS AND FUTURE RESEARCH

12.1. Introduction

The communication industries have been subject to much change in recent times. Two changes in particular have emphasised to a service provider the importance of setting appropriate processing resource levels. First, as competition grows, service development in multimedia applications like videoconferencing is likely to increase. These services require more processing resources to run, and as demand grows, processing can become a potential network bottleneck. Second, the development of dramatically improved routing protocols and technologies (such as fibre optics) has increased transportation capacities significantly. Both these factors have led some researchers to change their focus, and examine the networks from a processing capacity point of view.

A review of the literature (Chapter 2) highlights the need for further research into the processing capacity expansion decision. This thesis develops optimisation models describing aspects of processing capacity expansion, service provision, and distributed processing in modern communication networks. Unlike previous research, aspects

considered include delivery times, quality of service, and congestion; increasing model complexity greatly. The models in this thesis are developed under the assumptions that services are processing-based, and that transportation capacity is not a bottleneck. Significant research exists that studies the transportation capacity part of the network. Section 12.2 of this chapter summarises the main results and ideas presented in this thesis, and Section 12.3 outlines possible areas for future study.

12.2. Summary

The processing capacity expansion problem determines what levels of processing capacity to set in the network, considering the costs of providing capacity (investment, maintenance), and the benefits from providing it (processing decisions profits). Decisions are made subject to budget, demand, and ensuring acceptable quality of service is provided.

Because the costs of providing processing capacity are incurred immediately, they are easily determined. Conversely, the revenues from providing processing capacity are earned over time and are, therefore, uncertain. Because this revenue is earned at the operational level, it is important to study the operational problem itself.

Given fixed capacity, the operational processing decisions determine what (and how) demand would be met. Specifically, they decide which processing nodes to run which subservices on, considering, among other things, how demand will be met once those subservices are installed. One of the important strategic issues to come out of this investigation is to recognise that it is important to protect future profits at the operational level. Because future demand, and hence future profit, is harmed when customers receive inadequate quality of service, profits are protected in the model by ensuring adequate quality of service (measured by delivery times and rejections) is provided to customers.

An important operational issue was also detailed at this point; that being that congestion causes the marginal delivery times to increase as more demand is met and more capacity is used. This inherent convexity is modelled piece-wise linearly.

Implicitly considering queueing and congestion in this manner differs from the analytical formulae usually used to represent these queueing networks.

Analysis of the operational model found that it is this convexity that causes solutions with distributed processing to occur. Otherwise, centralised processing solutions occur. This finding explains why a high amount of operational model complexity is necessary in the capacity expansion model to obtain appropriate capacity solutions.

Whilst it is important to investigate the operational model in detail, this model drastically simplifies one of the most important factors to consider when capacity decision-making: demand. For the strategic capacity model, demand is much more complicated, and as such a deterministic estimate is inappropriate. Instead, it is important to develop a model of demand, one which reflects its complex nature.

To develop this model of demand it is first important to recognise that demand is implicitly dependent on capacity. With significant amounts of capacity, a large amount of demand can be met with acceptable delay, but, with less capacity, the network becomes congested faster, and hence less demand can be met with good quality of service. As far as we are aware, this thesis contains the first instance of capacity expansion models developed with demand being dependent on service levels and capacity. The literature generally models demand as independent of the capacity provided.

Demand in the capacity model is also complicated because it fluctuates from period to period, depending on how the quality of service provided compares with what customers expect. If the customers view the quality of service as being poor (relative to what they expect), then there will likely be a reduction in demand in the following period. This reduction occurs because customers become dissatisfied and reduce their own requests, as well as potentially reducing other customers' requests through negative word of mouth. Conversely, good quality of service leads to satisfied customers, and increased demand. Despite these period to period changes, an important result of this thesis is to show that demand could be modelled at a long-run level. This long-run level would be the point where the quality of service provided equals the level customers expect. Period to period changes will push demand to this

'equilibrium' level. This result is shown using a demand simulation model, which, for a given processing capacity, plays out how demand might react to the operational decisions made. While this simulation has limitations, it provides a reasonable representation of this demand process.

Our intention to investigate the processing capacity investment decisions while considering transportation and quality of service aspects has led to the development of a two-stage stochastic integer program (the PC1 model). The predominant difference between this model and those in the literature is the consideration of the complex relationship between capacity, quality of service, and demand.

Due to the combinatorial aspects of integer variables and the complexity that arises from uncertainty (scenarios), the developed model is highly complex. Hence, this thesis examines the impact of model approximations on solution quality. The demand simulation, discussed above, is used to give an idea of the degradation in solution quality brought about by model approximations. This illustrates which aspects of the model appear to be the most important.

Approximations were made to the capacity model's complexity, particularly to its integrality and uncertainty, and the results suggested, for the generated environment, the following guidelines:

- It appears important to include the stochasticity of the spread of demand parameter in the model. Conversely, the delivery times customers expect and the maximum demand pool parameters can be included at their deterministic averages.
- The amount of processing capacity used to install subservices can be approximated with a constant amount, which is estimated using model data.
- The second stage integer variables, required to include the piece-wise nature of the processing time function, and the integer aspect of the first stage variables, should remain.

This research shows that approximations can be made to the model to allow it to be solved in a reasonable time, whilst still producing a good quality of solution. Whilst

approximating complex model aspects can reduce solution times to a reasonable level, for large networks even these approximated models will have prohibitive solution times. Testing shows that, in particular, solution times increased dramatically as the number of potential processing locations increased. Initial testing shows that methods for approximating large networks could have some merit, but more thorough research is needed to confirm this.

12.3. Future Research

The models developed in this thesis have limitations. The remainder of this chapter summarises potential research avenues that could improve processing capacity expansion decision-making.

- For ease of understanding, models developed in this thesis are mixed-integer programs. Any non-linearity present in the problem situation is approximated. In order to improve model accuracy, and hence potentially model solutions, future research could include some of this non-linearity in the model. Of particular importance could be to treat demand for each service received at each location individually, not aggregately as is done in this thesis. Because of the non-convexity present, fine-tuned solution methods for these models may need to be developed.
- As the Internet becomes more and more stretched, more customers will be prepared to pay a premium for better service. We believe this extension (different classes of customers), discussed in Chapter 10, would be an important addition to the model.
- Despite being a simple model extension, considering multiple expansion points in the horizon could have important ramifications for decision-making. It is important to investigate whether decision-making is improved by expanding capacity over time.
- Future research could look at combining the processing capacity expansion and routing network design decisions; increasing model complexity greatly.

Melkote and Daskin (2001) investigate this problem, but they do not consider network congestion.

- Finally, in this thesis we have avoided prohibitive model solution times in our testing by making model approximations. Future research could investigate better ways of solving the PC1 model presented in Chapter 8, possibly using a decomposition approach. More efficient solution algorithms would eliminate the need for model approximation, or would allow the service provider to solve larger problems in the same amount of time.

REFERENCES

Adhikari, R. (1998), "Balance the Load on Web Servers", *Informationweek*, **674**, 109-116.

Agarwal, Y. K. (2002), "Design of Capacitated Multicommodity Networks with Multiple Facilities", *Operations Research*, **50** (2), 333-344.

Amiri, A. and Pirkul, H. (1999), "Routing and Capacity Assignment in Backbone Communication Networks under Time Varying Traffic Conditions", *European Journal of Operational Research*, **117** (1), 15-29.

Amiri, A., Rolland, E., and Barkhi, R. (1999), "Bandwidth Packing with Queuing Delay Costs: Bounding and Heuristic Solution Procedures", *European Journal of Operational Research*, **112** (3), 635-645.

Aneja, Y. P. and Chaouch, B. A. (1993), "Capacity Expansion and Contraction of a Facility with Economies of Scale", *Infor*, **31** (4), 247-260.

Audestad, J. A. (1998a), "Strategy for Enterprise Specifications", *Telektronikk*, **3/4.98**, 24-32.

Audestad, J. A. (1998b), "Telecommunications and Complexity", *Telektronikk*, **3/4.98**, 2-20.

Baentsch, M., Baum, L., Molter, G., Rothkugel, S., and Sturm, P. (1997), "World Wide Web Caching: The Application-Level View of the Internet", *IEEE Communications Magazine*, **35**, 170-178.

Balakrishnan, A., Magnanti, T. L., Shulman, A., and Wong, R. T. (1991), "Models for Planning Capacity Expansion in Local Access Telecommunications Networks", *Annals of Operations Research*, **33**, 239-284.

Barr, W. J., Boyd, T., and Inoue, Y. (1993), "The TINA Initiative", *IEEE Communications Magazine*, **31** (3), 70-76.

Barton, C. (2000), "Free Internet Offer Hooks Avid Browsers", New Zealand Herald, March 7.

Berman, O. and Ganz, Z. (1994), "The Capacity Expansion Problem in the Service Industry", *Computers and Operations Research*, **21** (5), 557-572.

Bienstock, D. and Shapiro, J. F. (1988), "Optimizing Resource Acquisition Decisions by Stochastic Programming", *Management Science*, **34** (2), 215-229.

Boorstyn, R. and Frank, H. (1977), "Large Scale Network Topological Optimization", *IEEE Transactions on Communications*, **COM-25**, 29-47.

Boucherie, R. J. and Van Dijk, N. M. (2000), "On a Queueing Network Model for Cellular Mobile Telecommunications Networks", *Operations Research*, **48** (1), 38-49.

Brauers, J. and Weber, M. (1988), "A New Method of Scenario Analysis for Strategic Planning", *Journal of Forecasting*, **7**, 31-47.

Buss, A. H., Lawrence, S. R., and Kropp, D. H. (1994), "Volume and Capacity Interaction in Facility Design", *IIE Transactions*, **26** (4), 36-49.

Calvert, K. L., Doar, M. B., and Zegura, E. W. (1997), "Modeling Internet Topology", *IEEE Communications Magazine*, **35**, 160-163.

Carpenter, B. E. and Kandlur, D. D. (1999), "Diversifying Internet Delivery", *IEEE Spectrum*, **36** (11), 57-61.

Chand, S., McClurg, T., and Ward, J. (2000), "A Model for Parallel Machine Replacement with Capacity Expansion", *European Journal of Operational Research*, **121** (3), 519-531.

Chang, S.-G. and Gavish, B. (1993), "Telecommunications Network Topological Design and Capacity Expansion: Formulations and Algorithms", *Telecommunications Systems*, **1**, 99-131.

Chang, S.-G. and Gavish, B. (1995), "Lower Bounding Procedures for Multiperiod Telecommunications Network Expansion Problems", *Operations Research*, **43** (1), 43-57.

Chapman, A. and Kung, H. T. (1998), "Enhancing Transport Networks with Internet Protocols", *IEEE Communications Magazine*, **36**, 100-104.

Chen, Z.-L., Li, S., and Tirupati, D. (2002), "A Scenario-Based Stochastic Programming Approach for Technology and Capacity Planning", *Computers and Operations Research*, **29** (7), 781-806.

Christofides, N. and Brooker, P. (1974), "Optimal Expansion of an Existing Network", *Mathematical Programming*, **6**, 197-211.

Chu, K. and Altmann, J. (2000), "Demand for Different Qualities of Service for Internet Access: A Review of Index Findings", *Philosophical Transactions of the Royal Society of London*, **358**, 2319-2334.

Cleveland, W. S. and Sun, D. X. (2000), "Internet Traffic Data", *Journal of the American Statistical Association*, **95** (451), 979-985.

Courcoubetis, C., Kelly, F., and Weber, R. (2000), "Measurement-Based Usage Charges in Communications Networks", *Operations Research*, **48** (4), 535-548.

Cowie, J. H., Nicol, D. M., and Ogielski, A. T. (1999), "Modeling the Global Internet", *Computing in Science & Engineering*, **1** (1), 42-50.

Cox, J. (1997), "Internet Service Provider Ensures Response Times", *Network World*, **14** (1), 10.

Crowcroft, J. (2000), "Herding Cats: Modelling Quality of Service for Internet Applications", *Philosophical Transactions of the Royal Society of London*, **358**, 2209-2215.

Daellenbach, H. G. (2001), *Systems Thinking and Decision Making: A Management Science Approach*, REA Publications, Christchurch, New Zealand.

Dasci, A. and Verter, V. (2001), "The Plant Location and Technology Acquisition Problem", *IIE Transactions*, **33**, 963-973.

Dewan, S. and Mendelson, H. (1990), "User Delay Costs and Internal Pricing for a Service Facility", *Management Science*, **36** (12), 1502-1517.

Dixit, A. K. and Pindyck, R. S. (1994), *Investment under Uncertainty*, Princeton University Press, Princeton.

Dowd, K. (1997), "The Best Routes for Net Links", *Informationweek*, **617**, 53-59.

Dryden, P. (1996), "Dodging 'Net Saturation", *Computerworld*, **30** (40), 55-56.

Duffy, J. (1998), "New Ways of Routing the Internet", *Network World*, **15** (3), 20.

Dutta, A. and Kim, Y. K. (1996), "A Heuristic Approach for Capacity Expansion of Packet Networks", *European Journal of Operational Research*, **91** (2), 395-410.

Dutta, A. and Lim, J.-I. (1992), "A Multiperiod Capacity Planning Model for Backbone Computer Communication Networks", *Operations Research*, **40** (4), 689-705.

Dye, S., Tomasgard, A., and Wallace, S. W. (1998), "New Aspects of Service Provision and Technology Strategies in Telecommunications", *Telektronikk*, **3/4.98**, 67-73.

Dyer, M. and Stougie, L. (1996), "Stochastic Programming Problems: Complexity and Approximability", Working Paper, Eindhoven University of Technology.

Eppen, G. D., Martin, R. K., and Schrage, L. (1989), "A Scenario Approach to Capacity Planning", *Operations Research*, **37** (4), 517-527.

Erlenkotter, D. (1967), "Two Producing Areas - Dynamic Programming Solutions", in Manne, A. S., ed.: *Investments for Capacity Expansion: Size, Location and Time Phasing*, MIT Press, Cambridge, Massachusetts.

Erlenkotter, D. (1973), "Sequencing Expansion Projects", *Operations Research*, **21** (2), 542-553.

Evans, J. R. (1997), *Production/Operations Management: Quality, Performance, and Value*, West Publishing Company, St. Paul.

Fine, C. H. and Freund, R. M. (1990), "Optimal Investment in Product-Flexible Manufacturing Capacity", *Management Science*, **36** (4), 449-466.

Flippo, O. E., Kolen, A. W. J., Koster, A. M. C. A., and Van De Leensel, R. L. M. J. (2000), "A Dynamic Programming Algorithm for the Local Access Telecommunication Network Expansion Problem", *European Journal of Operational Research*, **127** (1), 189-202.

Fourer, R., Gay, D. M., and Kernighan, B. W. (1993), *AMPL: A Modelling Language for Mathematical Programming*, The Scientific Press.

Fowler, T. B. and Wright, K. J. (1994), "Telecommunications for the 21st Century", *OR/MS Today*, **June**, 20-27.

Fratta, L., Gerla, M., and Kleinrock, L. (1973), "The Flow-Deviation Algorithm: An Approach to Store-and-Forward Computer Communication Network Design", *Networks*, **3**, 97-133.

Freidenfelds, J. (1981), *Capacity Expansion: Analysis of Simple Models with Applications*, North Holland, Amsterdam.

Gaimon, C. and Ho, J. C. (1994), "Uncertainty and the Acquisition of Capacity: A Competitive Analysis", *Computers and Operations Research*, **21** (10), 1073-1089.

Garcia, B.-L., Mahey, P., and Le Blanc, L. J. (1998), "Iterative Improvement Methods for a Multiperiod Network Design Problem", *European Journal of Operational Research*, **110** (1), 150-165.

Gartner, B. and Nelson, B. (1998), "If You Build It, They Will Come", *Telephony*, **234** (23), 152-158.

Gavish, B. (1982), "Topological Design of Centralised Computer Networks", *Networks*, **12**, 355-377.

Gavish, B. (1992), "Topological Design of Computer Communication Networks - the Overall Design Problem", *European Journal of Operational Research*, **58** (2), 149-172.

Gavish, B. and Neuman, I. (1989), "A System for Routing and Capacity Assignment in Computer Communication Networks", *IEEE Transactions on Communications*, **37** (4), 360-366.

Gerla, M. (1973), "Deterministic and Adaptive Routing Policies in Packet Switched Communication Networks", *Data Networks: Analysis and Design - 3rd Data Communications Symposium*, 23-28.

Gerla, M. and Kleinrock, L. (1977), "On the Topological Design of Distributed Computer Networks", *IEEE Transactions on Communications*, **COM-25**, 48-60.

Graham, J. (1991), *Dictionary of Telecommunications*, Penguin, London.

Higgins, R. and Woods, R. (1998), "The Business of the Internet", *Telecommunications International Edition*, **32** (4), 39-44.

Hiller, R. S. and Shapiro, J. F. (1986), "Optimal Capacity Expansion Planning When There Are Learning Effects", *Management Science*, **32** (9), 1153-1163.

Hoyland, K. and Wallace, S. W. (2001), "Generating Scenario Trees for Multistage Decision Problems", *Management Science*, **47** (2), 295-307.

Huberman, B. A. and Lukose, R. M. (1997), "Social Dilemmas and Internet Congestion", *Science*, **277**, 535-537.

Ierapetritou, M. G. and Pistikopoulos, E. N. (1994), "Novel Optimization Approach of Stochastic Planning Models", *Industrial and Engineering Chemistry Research*, **33**, 1930-1942.

Inoue, Y., Guha, D., and Berndt, H. (1998), "The TINA Consortium", *IEEE Communications Magazine*, **36** (9), 130-136.

IRN (2000), "ISP Suspends New Customer Registrations", New Zealand Herald, December 18.

Jack, C., Kai, S. R., and Shulman, A. (1992), "Design and Implementation of an Interactive Optimization System for Telephone Network Planning", *Operations Research*, **40** (1), 14-25.

Jaskold-Gabszewicz, J. and Vial, J. P. (1972), "Optimal Capacity Expansion under Growing Demand and Technological Progress", in Szego, G. P. and Shell, K., eds.: *Mathematical Methods in Investment and Finance*, North-Holland Publishers, Amsterdam.

Kall, P. and Wallace, S. W. (1994), *Stochastic Programming*, John Wiley & Sons, Chichester.

Karmarkar, U. and Kekre, S. (1987), "Manufacturing Configuration, Capacity and Mix Decisions Considering Operational Costs", *Journal of Manufacturing Systems*, **6** (4), 315-324.

Kelly, F. P. (2000), "Models for a Self-Managed Internet", *Philosophical Transactions of the Royal Society of London*, **358**, 2335-2348.

Kim, D. and Lee, W. J. (1998), "Optimal Joint Pricing and Lot Sizing with Fixed and Variable Capacity", *European Journal of Operational Research*, **109** (1), 212-227.

Kitson, B., Lewis, I., and Parhar, A. (1994), "PLATyPus: Experimenting with TINA", *IEEE Communications Magazine*, **32** (12), 72-79.

Klassen, K. J. and Rohleder, T. R. (2001), "Combining Operations and Marketing to Manage Capacity and Demand in Services", *The Services Industries Journal*, **21** (2), 1-30.

Klincewicz, J. G. (1994), "Transmit and Transport", *OR/MS Today*, **June**, 30-39.

Kreidi, A. and Sanso, B. (1994), "Optimization of a Geographically Distributed Air-Ground Airline Telecommunication System", Technical Report, GERAD Publication G-94-47.

Laguna, M. (1998), "Applying Robust Optimization to Capacity Expansion of One Location in Telecommunications with Demand Uncertainty", *Management Science*, **44** (11), S101-S110.

Lawson, S. (1996), "Internet Gridlock Escalates", *InfoWorld*, **18** (29), 1,24.

Le Blanc, L. J., Chifflet, J., and Mahey, P. (1999), "Packet Routing in Telecommunication Networks with Path and Flow Restrictions", *INFORMS Journal on Computing*, **11** (2), 188-197.

Lee, B. P., Balan, R. K., Jacob, L., Seah, W. K. G., and Ananda, A. L. (2002), "Avoiding Congestion Collapse on the Internet Using TCP Tunnels", *Computer Networks*, **39**, 207-219.

Lee, Y., Kim, S.-I., Han, J., and Kang, K. (2001), "An ATM Switch Capacity Allocation Problem", *Telecommunication Systems*, **18** (4), 301-313.

Lewis, P. H. (1996), "New Flat Rate Creates Surge in Use of America Online", New York Times, December 3.

Li, S. and Tirupati, D. (1994), "Dynamic Capacity Expansion Problem with Multiple Products: Technology Selection and Timing of Capacity Additions", *Operations Research*, **42** (5), 958-976.

Li, S. and Tirupati, D. (1995), "Technology Choice with Stochastic Demands and Dynamic Capacity Allocation: A Two Way Product Analysis", *Journal of Operations Management*, **12** (3), 239-258.

Lind, J. (1998), "Finding the Fastest Path", *Communications News*, **35** (2), 71.

Little, J. D. C. (1961), "A Proof of the Queueing Formula L = 'Lambda'w", *Operations Research*, **9** (3), 383-387.

Lohr, S. (1997), "Refunds Planned by America Online in Network Jam", New York Times, January 30.

Louveaux, F. V. (1986), "Discrete Stochastic Location Models", *Annals of Operations Research*, **6**, 23-34.

Lundin, R. A. and Morton, T. E. (1975), "Planning Horizons for the Dynamic Lot Size Model: Zabel Versus Protective Procedures and Computational Results", *Operations Research*, **23** (4), 711-734.

Luss, H. (1982), "Operations Research and Capacity Expansion Problems: A Survey", *Operations Research*, **30** (5), 907-947.

Magnanti, T. L. and Wong, R. T. (1984), "Network Design and Transportation Planning: Models and Algorithms", *Transportation Science*, **18**, 1-55.

Manne, A. S. (1961), "Capacity Expansion and Probabilistic Growth", *Econometrica*, **29**, 632-649.

Manne, A. S. (1967), *Investments for Capacity Expansion: Size, Location and Time-Phasing*, MIT Press, Cambridge, Massachusetts.

Melkote, S. and Daskin, M. S. (2001), "Capacitated Facility Location/Network Design Problems", *European Journal of Operational Research*, **129** (3), 481-495.

Messerschmitt, D. G. (1996), "The Convergence of Telecommunications and Computing: What Are the Implications Today?" *Proceedings of the IEEE*, **84** (8), 1167-1186.

Miller, P. (1997), *TCP/IP Explained*, Digital Press, Boston.

Minoux, M. (1987), "Network Synthesis and Dynamic Network Optimization", *Annals of Discrete Mathematics*, **31**, 283-324.

Mookerjee, V. S. and Tan, Y. (2002), "Analysis of a Least Recently Used Cache Management Policy for Web Browsers", *Operations Research*, **50** (2), 345-357.

Mullender, S. (1993), *Distributed Systems*, Addison-Wesley, New York.

Nain, P. and Ross, K. W. E. (1992), "Stochastic Modelling of Telecommunications Systems", *Annals of Operations Research*, **35**.

Neebe, A. W. and Rao, M. R. (1986), "Sequencing Capacity Expansion Projects in Continuous Time", *Management Science*, **32** (11), 1467-1479.

Rajagopalan, S. (1992), "Deterministic Capacity Expansion under Deterioration", *Management Science*, **38** (4), 525-539.

Rajagopalan, S. (1994), "Capacity Expansion with Alternative Technology Choices", *European Journal of Operational Research*, **77** (3), 392-403.

Rajagopalan, S. (1998), "Capacity Expansion and Equipment Replacement: A Unified Approach", *Operations Research*, **46** (6), 846-857.

Rajagopalan, S., Singh, M. R., and Morton, T. E. (1998), "Capacity Expansion and Replacement in Growing Markets with Uncertain Technological Breakthroughs", *Management Science*, **44** (1), 12-30.

Rajagopalan, S. and Soteriou, A. C. (1994), "Capacity Acquisition and Disposal with Discrete Facility Sizes", *Management Science*, **40** (7), 903-917.

Rajagopalan, S. and Yu, H.-L. (2001), "Capacity Planning with Congestion Effects", *European Journal of Operational Research*, **134** (2), 365-377.

Roberts, J. W. (2001), "Traffic Theory and the Internet", *IEEE Communications Magazine*, **39** (1), 94-99.

Sampat, P. (2000), "Internet Use Accelerates", in Brown, L. R., Renner, M. and Halweil, B., eds.: *Vital Signs 2000: The Environmental Trends That Are Shaping Our Future*, W.W. Norton & Company, New York.

Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., and Zahorjan, J. (1999), "Detour: Informed Internet Routing and Transport", *IEEE Micro*, **19** (1), 50-59.

Sen, S., Doverspike, R. D., and Cosares, S. C. (1992), "Network Planning with Random Demand", *Telecommunications Systems*, **3**, 11-30.

Sethi, S. P., Taksar, M., and Zhang, Q. (1995), "Hierarchical Capacity Expansion and Production Planning Decisions in Stochastic Manufacturing Systems", *Journal of Operations Management*, **12** (3,4), 331-352.

Shotsberger, P. G. (1996), "Toward a Greener Internet", *Computer*, **29**, 113-114.

Simampo, A. and Ryan, S. M. (2001), "Capacity Expansion for a Loss System with Exponential Demand Growth", Working Paper, Iowa State University.

Simon, H. A. and Ando, A. (1961), "Aggregation of Variables in Dynamic Systems", *Econometrica*, **29**, 111-138.

Smith, J. E. and McCardle, K. F. (1999), "Options in the Real World: Lessons Learned in Evaluating Oil and Gas Investments", *Operations Research*, **47** (1), 1-15.

So, K. C. and Song, J. S. (1998), "Price, Delivery Time Guarantees and Capacity Selection", *European Journal of Operational Research*, **111** (1), 28-49.

Sridharan, R. (1995), "The Capacitated Plant Location Problem", *European Journal of Operational Research*, **87** (2), 203-213.

Stallings, W. (1995), *ISDN and Broadband ISDN with Frame Relay and ATM*, Prentice Hall, New Jersey.

Stougie, L. and Van Der Vlerk, M. H. (1997), "Stochastic Integer Programming", in Dell'amico, M., Maffioli, F. and Martello, S., eds.: *Annotated Bibliographies in Combinatorial Optimization*, Wiley, New York.

Stuart, F. I. and Tax, S. S. (1996), "Planning for Service Quality: An Integrative Approach", *International Journal of Service Industry Management*, **7** (4), 58-77.

Taylor, S. (1994), "Waiting for Service: The Relationship between Delays and Evaluations of Service", *Journal of Marketing*, **58** (2), 56-69.

Tomasgard, A. (1998), *Aspects of Service Provision and Distributed Processing in a Telecommunication Network*, Ph.D. Thesis, Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology.

Tomasgard, A., Audestad, J. A., Dye, S., Stougie, L., Van Der Vlerk, M. H., and Wallace, S. W. (1998), "Modelling Aspects of Distributed Processing in Telecommunication Networks", *Annals of Operations Research*, **82**, 161-184.

Tomasgard, A., Dye, S., Wallace, S. W., Audestad, J. A., Stougie, L., and Van Der Vlerk, M. H. (1997), "Stochastic Optimization Models for Distributed Communications Networks", Working Paper no. 3/97, Norwegian University of Science and Technology.

Van Der Vlerk, M. H. (2002), "Stochastic Integer Programming Bibliography", *World Wide Web, http://mally.eco.rug.nl/biblio/stoprog.html.*

Wallace, S. W. (1998), "The Role of Uncertainty in Strategic Planning", *Telektronikk*, **3/4.98**, 21-23.

Wirth, P. E. (1997), "The Role of Teletraffic Modeling in the New Communications Paradigms", *IEEE Communications Magazine*, **35**, 86-92.

APPENDICES

A. MODEL TESTING

A.1. Testing Long-Run Demand Approximation

For a given data set (see Appendix B for how these were chosen), LD was found from the PC1 model presented in Chapter 8 and the processing decisions period's average demand (AD) over the planning horizon was found from the RP model presented in Chapter 7. These were compared for 200 data sets, and the results are presented in Table A.1 and Figure A.1. The AMPL formulation for the RP model is in Appendix C.

With a maximum demand pool of 10 000, the initial demand was randomly sampled from a U(8000, 10 000) distribution. External factors were tested with high volatility (±20% ~ U(-2000, 2000)) and low volatility (±5% ~ U(-500, 500)) distributions. Parameters *fr_rejs* and *fr_dts* were tested with high and low rates of change, where higher rates of change mean the demand should go to its long-run level faster.

Average Demand Approximation				
PC1 model versus	**High rate of change**		**Low rate of change**	
	RP model (low volatility)	**RP model (high volatility)**	**RP model (low volatility)**	**RP model (high volatility)**
Mean Percentage Difference between LD and AD[21]	5.37%	2.40%	2.04%	2.34%
95% Confidence Interval of Percentage Differences	(0.58%, 6.94%)	(-0.59%, 5.25%)	(-0.37%, 3.25%)	(-3.37%, 8.39%)

Table A.1. Summary of percentage differences between PC1 model's LD and RP model's AD.

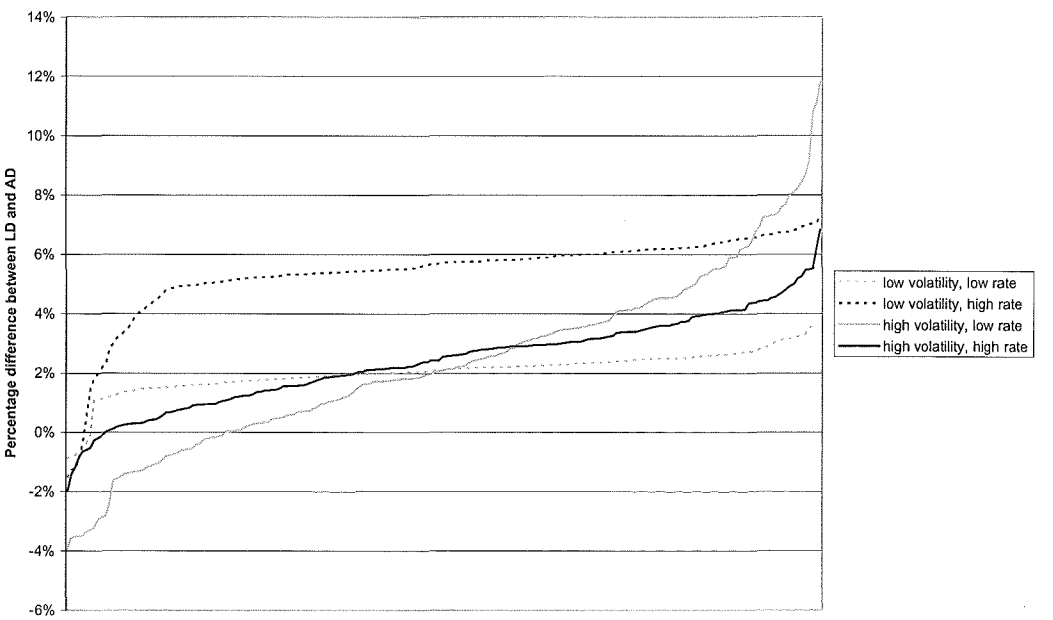


Figure A.1. Percentage differences between PC1 model's LD and RP model's AD.

[21] A positive difference indicates that LD was higher than AD.

As shown by these results, long-run demand appears to be a reasonable approximation of period-to-period average demand under high and low demand volatility.

B. DATA FOR MODEL RUNS

Some aspects of the PC1 model presented in Chapter 8 only add unnecessary complication to the model for testing purposes. These parameters are fixed for each model run; the remaining parameters are randomly generated each run.

B.1. Fixed-value Parameters

As they do not change the model's underlying structure, the following parameters only add unnecessary differences to the models in the testing process. Hence, they are fixed for all model runs.

- Arc capacity is unlimited ($q_a = 8, \forall a$). This is appropriate, as we are assuming transportation is not the bottleneck. There are effective capacities on arc usage because demand is limited due to processing capacities.

- Both the time and per-usage costs associated with routing over an arc are zero ($t_a = 0$ and $v_a = 0$, $\forall\ a$). The per-usage cost of using a routing node is also zero ($w_{rn} = 0$, $\forall\ rn$). Making these parameters non-zero would have little impact on solutions.
- Each service consists of two subservices, where one unit of each subservice is required when processing a request ($h_{k,s} = 1$, $\forall\ k,s$). This subservice unit requires one unit of flow to route it ($j_k = 1$, $\forall\ k$). These assumptions simplify the test problems.
- From Figure 5.2, there are no minimum addition levels ($l_{min} = 0$), and an infinite amount of processing capacity could be installed on each computer ($l_{max} = \infty$). Hence, there is only one computer per processing node.
- Processing capacity costs are fully incurred at the investment level. There is no per-usage cost for using processing capacity ($b_{c,n} = 0$, $\forall\ c,n$).
- There are no costs associated with maintaining a computer ($g_{c,n} = 0$, $\forall\ c,n$).
- There are no budgetary restrictions for capacity expansion ($z = \infty$).
- Each potential processing location is a demand received location.
- The discount factor is arbitrarily set to 5% per annum. Also, there are assumed to be 500 weekly processing decisions periods in the planning horizon. Hence, $\beta = \sum_{n=0}^{499} \frac{1}{1.001^n} = \left(\frac{1-\left(\frac{1}{1.001}\right)^{499}}{1-\frac{1}{1.001}}\right) = 393$.
- In order for the investment to be worthwhile undertaking it was assumed that the investment costs would be paid off after roughly 105 discounted processing decisions periods ($\beta = 100$) of the entire planning horizon. The revenue from selling the service and the investment costs were set to roughly ensure this 'rule of thumb' was met. If we set average fixed investment costs (*invf*) to be 1 000 000 and average variable investment costs (*invv*) to be 1550, then it follows (from the following equation) that the revenue from selling a

service should be about \$40 ($p_{s,dr} = 40$, $\forall$ *s,dr*). Note that, on average, demand (*d*) is 10 000, and five computers (*c*) are installed with 25 000 units of processing capacity (*pc*).

$$100\, p\, d = invf * c + invv * pc$$

$$\Rightarrow \qquad p = 40.$$

B.2. Randomly Generated Parameters

For each observation used in the testing process different values for the remaining parameters were generated. This section details how the uncertain parameters were randomly generated. The reader is referred to Appendix C for precise details on this.

- The amount of processing capacity it takes to install a subservice (u_k) is randomly sampled as being large (1000 units), medium (600), or small (200).
- The investment cost structure for a computer ($C_{c,n}(\ldots)$), or more precisely the ratio of the fixed to the variable investment costs is randomly determined for each run. This was done by keeping the fixed cost at its average (1 000 000), and varying the variable investment cost around its average (U(100, 3000)). Ratios change from run to run to reflect that some processing capacity investment structures will be predominantly fixed cost based, whilst others will be variable cost based.
- The processing time functions ($F_{c,n}(\ldots)$) always have the same shape, based on the description provided by Le Blanc *et al* (1999). However, for each run the actual curve is randomly chosen, while retaining this shape, as shown in Figure B.1. Also, all functions are the same at different nodes.

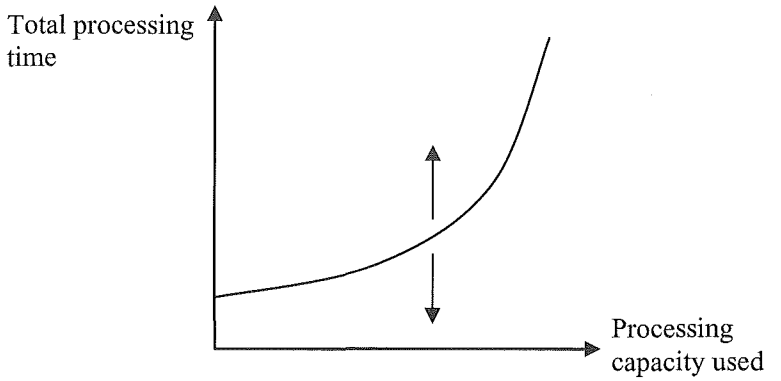


Figure B.1. Processing time function shape for model runs.

- The transportation time functions ($H_{rn}(\ldots)$) have the same shape as above, but are also randomly chosen (independently from the processing time function) for each run. It is assumed that a service spends, on average, the same amount of time being routed as it does processed.
- The average delivery times customers expect (*ce*) are assumed to come from a N (μ=7.5, σ=1) distribution. The average rejections customers expect (*re*) are assumed to be zero. This is because customers are highly adverse to being rejected.
- The maximum demand pool (*d*) is assumed to come from a N (μ=10 000, σ=1000) distribution.
- The underlying base for the spread of demand is randomly generated for each run. This is done by randomly choosing whether each demand location is big, medium, or small (with equal probability). The spread of demand parameter is generated for each scenario as discussed in Chapter 11.

C. AMPL FILES

C.1. Introduction

Contained in this appendix are the model, data, and run files, formulated in AMPL, for the PC1 model with spread of demand stochasticity. All the possible integer approximations (outlined in Chapter 11) are included. Following that, the AMPL files for the RP model (the demand simulation model) are presented.

Of particular interest, Section C.2 shows the AMPL formulation of the PC1 model developed in Chapter 8, including the explicit formulation of the processing and transportation time functions. Section C.6 presents the data used for the RP model, specifically the rate of change functions shown in Chapter 7. The explicit method for how demand changes from period to period is presented in Section C.7. Finally, Section C.3 shows how the spread of demand scenarios are sampled.

C.2. PC1 Model File

```
# pc1.mod

                                problem pc1Model;
# ======================================================================

# Sets

param num_pns >= 0;
set PROC_NODES := {1..num_pns};
param num_drnodes >= 0;
set DR_NODES := {1..num_drnodes};
param max_services >= 0 integer, := 10;
set MAX_SERVICES := {1..max_services};
param num_services >= 0 integer, <= max_services;
set SERVICES := {1..num_services};
param max_subs >= 0 integer, := 10;
set MAX_SUBS := {1..max_subs};
param num_subs >= 0 integer, <= max_subs;
set SUBS := {1..num_subs};
param num_rns >= 0;
set ROUTING_NODES := {1.. num_rns};
set SUBSFOR {MAX_SUBS};
set ROUTES {DR_NODES,PROC_NODES};
set ROUTES_THRU_NODES {ROUTING_NODES}
      within {dr in DR_NODES,n in PROC_NODES,ROUTES[dr,n]};
param max_scenarios >= 0 integer, := 100;
set MAX_SCENARIOS := {1..max_scenarios};
param num_scenarios >= 0 integer, <= max_scenarios;
set SCENARIOS := {1..num_scenarios};

# Parameters

param ip_lp binary;
# 0 if IP, 1 if LP
param nt_pn binary;
# 0 if network estimation, 1 if processing node specific estimation
```

```
param coeff1_Nt := 2.4;
param coeff1_Pn := 1.8;
param coeff2_Pn := 0.015;
param num_periods >= 0;
param disc_factor >= 0;
param min_add {PROC_NODES} >= 0;
param maint_cost {PROC_NODES} >= 0;
param invest_fixed {PROC_NODES} >= 0;
param invest_vble {PROC_NODES} >= 0;
param invest_limit_unit {PROC_NODES} >= 0;

param cust_exp >= 0;
param exp_rejs >= 0, <= 1;
param probab_scen {MAX_SCENARIOS} >= 0, <= 1;
param large_number >= 0;
param fr_sdr {MAX_SERVICES,DR_NODES,MAX_SCENARIOS} >= 0;
param price {MAX_SERVICES,DR_NODES} >= 0;
param rej_cost {MAX_SERVICES,DR_NODES};
param sub_conversion {MAX_SUBS} >= 0;
param flow_conversion {MAX_SUBS} >= 0;
param installedsub_usage {MAX_SUBS} >= 0;
param node_cost {PROC_NODES} >= 0;
param service_use {MAX_SUBS,MAX_SERVICES} >= 0;
param node_route_cost {ROUTING_NODES} >= 0;
param npiece_ptime {PROC_NODES} integer >= 1;
param rate_ptime {n in PROC_NODES,q in 1..npiece_ptime[n]}
                  >= if q = 1 then 0 else rate_ptime[n,q-1] ;
param ptime_bkpt_fr {n in PROC_NODES,1..npiece_ptime[n]} >= 0;
param npiece_ttime {ROUTING_NODES} integer >= 1;
param rate_ttime {rn in ROUTING_NODES,q in 1..npiece_ttime[rn]}
                  >= if q = 1 then 0 else rate_ttime[rn,q-1] ;
param limit_ttime {rn in ROUTING_NODES,q in 1..npiece_ttime[rn]-1}
                  > if q = 1 then 0 else limit_ttime[rn,q-1] ;
param maxm_dpool >= 0;
param av_dt {MAX_SCENARIOS} >= 0;

param 2nd_stage_ints binary;
# 0 integer (piece-wise linear processing time function),
```

```
# 1 linear (linear processing time function)

param 1st_stage_ints binary;
# 0 integer (integer 1st stage variables),
# 1 linear (linear 1st stage variables)

param rate_ptime_linear >= 0;
# the weighted average of potential segments

# Decision Variables

var num_invest_unit {PROC_NODES} >= 0 integer;
var pc_loc {PROC_NODES} binary;
# integer element removed with 1st_stage_ints = 1
var num_invest_unit2 {PROC_NODES} >= 0;
var pc_loc2 {PROC_NODES} >= 0, <= 1;
var pc_add {PROC_NODES} >= 0;
var investment_costs >= 0;

var pds_level_profit {SCENARIOS};
var Total_dpool {SCENARIOS};
var Demand {SERVICES,DR_NODES,SCENARIOS} >= 0;
var demand_met {SERVICES,DR_NODES,SCENARIOS} >= 0;
var demand_rejected {SERVICES,DR_NODES,SCENARIOS} >= 0;
var demand_met_node {SUBS,DR_NODES,PROC_NODES,SCENARIOS} >= 0;
var sub_installed {SUBS,PROC_NODES,SCENARIOS} binary;
# integer element removed with ip_lp = 1
var routing {SUBS,dr in DR_NODES,n in PROC_NODES,ROUTES[dr,n],
                                          SCENARIOS} >= 0;
var delivery_costs {SCENARIOS} >= 0;
var total_pc_used_demand {PROC_NODES,SCENARIOS} >= 0;
var total_pc_used_subinstall {PROC_NODES,SCENARIOS} >= 0;
var total_pc_used {PROC_NODES,SCENARIOS} >= 0;
var proc_costs {SCENARIOS} >= 0;
var proc_time {PROC_NODES,SCENARIOS} >= 0;
var delivery_time {SCENARIOS} >= 0;
var transp_time {SCENARIOS} >= 0 ;
var time_through_node {ROUTING_NODES,SCENARIOS} >= 0;
```

```
var dm_flow_node {ROUTING_NODES,SCENARIOS} >= 0;
var pc_segment_used {n in PROC_NODES,1..npiece_ptime[n],SCENARIOS} >=
0;
var pc_segment_used_subs {n in
PROC_NODES,1..npiece_ptime[n],SCENARIOS} >= 0;
var pc_segment_used_dem {n in
PROC_NODES,1..npiece_ptime[n],SCENARIOS} >= 0;
var av_pc_subs {PROC_NODES} >= 0;

var z1 {n in PROC_NODES,1..npiece_ptime[n],SCENARIOS} binary;
var z2 {n in PROC_NODES,1..npiece_ptime[n],SCENARIOS} binary;
# binary variables required to explicitly consider the processing
# time function properly.
# integer element removed with 2nd_stage_ints = 1

# Objective Function

maximize long_run_profit:

      sum {sc in SCENARIOS} probab_scen[sc] *
            pds_level_profit[sc] * num_periods

      - investment_costs

      - sum {n in PROC_NODES} maint_cost[n] *
                  (num_invest_unit[n] * (1 - 1st_stage_ints) +
                  num_invest_unit2[n] * 1st_stage_ints);

# Constraints

subject to Investment {n in PROC_NODES}:
      pc_add[n] <= large_number * (pc_loc[n] * (1 - 1st_stage_ints) +
                                    pc_loc2[n] * 1st_stage_ints);

subject to Min_additions {n in PROC_NODES}:
      pc_add[n] >= invest_limit_unit[n]
                  * ((num_invest_unit[n] * (1 - 1st_stage_ints) +
                  num_invest_unit2[n] * 1st_stage_ints) - 1)
```

```
                + min_add[n];

subject to Computer_size {n in PROC_NODES}:
      pc_add[n] <= invest_limit_unit[n] *
                  (num_invest_unit[n] * (1 - 1st_stage_ints) +
                  num_invest_unit2[n] * 1st_stage_ints);

subject to Calculating_Investment_Costs:
      investment_costs = sum {n in PROC_NODES}
                        (invest_fixed[n] *
                        (num_invest_unit[n] * (1 - 1st_stage_ints) +
                        num_invest_unit2[n] * 1st_stage_ints) +
                        invest_vble[n] * pc_add[n]);

subject to Calculating_pds_level_profits {sc in SCENARIOS}:
      pds_level_profit[sc] = sum {s in SERVICES,dr in DR_NODES}
            (demand_met[s,dr,sc] * price[s,dr]
            - demand_rejected[s,dr,sc] * rej_cost[s,dr])

      - proc_costs[sc]

      - delivery_costs[sc] ;

subject to Find_proc_costs {sc in SCENARIOS}:
      proc_costs[sc] = sum {n in PROC_NODES}
                        total_pc_used[n,sc] * node_cost[n];

subject to Processing_Capacity_Constraint {n in PROC_NODES,
                                           sc in SCENARIOS}:
      total_pc_used[n,sc] <= pc_add[n] ;

subject to Find_total_pc_used {n in PROC_NODES,sc in SCENARIOS}:
      total_pc_used[n,sc] = total_pc_used_demand[n,sc]
                        + total_pc_used_subinstall[n,sc] ;

subject to Find_pc_used_for_demand {n in PROC_NODES,sc in SCENARIOS}:
      total_pc_used_demand[n,sc] = sum {k in SUBS,dr in DR_NODES}
                  demand_met_node[k,dr,n,sc] * sub_conversion[k];
```

```
subject to Find_av_pc_subs {n in PROC_NODES,sc in SCENARIOS}:
      av_pc_subs[n] =
                  (1 - nt_pn) * ((pc_loc[n] * (1 - 1st_stage_ints) +
                  pc_loc2[n] * 1st_stage_ints) * coeff1_Nt
                  * sum {k in SUBS} installedsub_usage[k] / num_pns)
                  + nt_pn * ((pc_loc[n] * (1 - 1st_stage_ints) +
                  pc_loc2[n] * 1st_stage_ints) * coeff1_Pn
                  * sum {k in SUBS} installedsub_usage[k] / num_pns
                  + coeff2_Pn * pc_add[n]);

subject to Find_pc_used_for_subinstall {sc in SCENARIOS,
                                        n in PROC_NODES}:
      total_pc_used_subinstall[n,sc] = av_pc_subs[n] * ip_lp
            + (1 - ip_lp) *
            sum {k in SUBS} sub_installed[k,n,sc]
                        * installedsub_usage[k];

subject to Finding_Pc_breakpoints {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      pc_segment_used[n,q,sc] <= ptime_bkpt_fr[n,q] * pc_add[n];

subject to Finding_Pc_breakpoints_b {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      pc_segment_used[n,q,sc] =
      pc_segment_used_subs[n,q,sc] + pc_segment_used_dem[n,q,sc];

subject to Summing_segments {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 0}:
      sum {q in 1..npiece_ptime[n]}
            pc_segment_used[n,q,sc] = total_pc_used[n,sc];

subject to Summing_segments_subs {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 0}:
      sum {q in 1..npiece_ptime[n]}
            pc_segment_used_subs[n,q,sc] =
                  total_pc_used_subinstall[n,sc];
```

```
subject to Summing_segments_demand {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 0}:
      sum {q in 1..npiece_ptime[n]}
            pc_segment_used_dem[n,q,sc] =
                  total_pc_used_demand[n,sc];

subject to Setting_dem_segments_1 {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      total_pc_used_subinstall[n,sc] - ptime_bkpt_fr[n,q] * pc_add[n]
            <= large_number * (1 - z1[n,q,sc]);

subject to Setting_dem_segments_2 {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      pc_segment_used_dem[n,q,sc] <= large_number * z1[n,q,sc];

subject to Setting_subs_segments_1 {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      ptime_bkpt_fr[n,q] * pc_add[n] - total_pc_used_subinstall[n,sc]
            <= large_number * (1 - z2[n,q,sc]);

subject to Setting_subs_segments_2 {n in PROC_NODES,sc in SCENARIOS,
      q in 1..npiece_ptime[n]:q < npiece_ptime[n] and
                                    2nd_stage_ints = 0}:
      pc_segment_used_subs[n,q+1,sc] <= large_number * z2[n,q,sc];

subject to Find_proc_time {n in PROC_NODES,sc in SCENARIOS}:
      proc_time[n,sc] = (sum {q in 1..npiece_ptime[n]}
                  rate_ptime[n,q] * pc_segment_used[n,q,sc]
                   - sum {q in 1..npiece_ptime[n]}
                  rate_ptime[n,q] * pc_segment_used_subs[n,q,sc])
            * (1 - 2nd_stage_ints)
                  + rate_ptime_linear * total_pc_used_demand[n,sc]
            * 2nd_stage_ints;

subject to Effective_cap {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 1}:
      total_pc_used[n,sc] <= (1 - ptime_bkpt_fr[n,5]) * pc_add[n];
# an 'effective capacity' in the case of linear processing time
```

```
# functions

subject to Find_dm_flow_node {rn in ROUTING_NODES,sc in SCENARIOS}:
      dm_flow_node[rn,sc]=
            sum {k in SUBS,dr in DR_NODES,n in PROC_NODES,
                        r in ROUTES[dr,n]:
                  (dr,n,r) in ROUTES_THRU_NODES[rn]}
                              routing[k,dr,n,r,sc] ;

subject to Find_routing_time {rn in ROUTING_NODES,sc in SCENARIOS}:
      time_through_node[rn,sc] =
      << {q in 1..npiece_ttime[rn]-1} limit_ttime[rn,q];
            {q in 1..npiece_ttime[rn]} rate_ttime[rn,q] >>
                        dm_flow_node[rn,sc];

subject to Find_transp_time {sc in SCENARIOS}:
      transp_time[sc] = sum {rn in ROUTING_NODES}
                              time_through_node[rn,sc];

subject to Find_delivery_time {sc in SCENARIOS}:
      delivery_time[sc] = sum {n in PROC_NODES}
                              proc_time[n,sc] + transp_time[sc] ;

subject to Find_delivery_costs {sc in SCENARIOS}:
      delivery_costs[sc] =
           sum {rn in ROUTING_NODES} node_route_cost[rn] *
              sum {k in SUBS, (ddr,nn,rr) in ROUTES_THRU_NODES[rn]}
                  routing[k,ddr,nn,rr,sc];

subject to Meeting_cust_exp_dts {sc in SCENARIOS}:
      delivery_time[sc] <= cust_exp * Total_dpool[sc];

subject to Meeting_cust_exp_rejs {sc in SCENARIOS}:
      sum {s in SERVICES,dr in DR_NODES} demand_rejected[s,dr,sc]
            <= exp_rejs * Total_dpool[sc];

subject to Maxm_Demand_Pool {sc in SCENARIOS}:
      Total_dpool[sc] <= maxm_dpool;
```

```
subject to Finding_indiv_demand {s in SERVICES,dr in DR_NODES,
                                          sc in SCENARIOS}:
      Demand[s,dr,sc] = fr_sdr[s,dr,sc] * Total_dpool[sc] ;

subject to Meet_Demand {s in SERVICES,dr in DR_NODES,
                                    sc in SCENARIOS}:
      Demand[s,dr,sc] = demand_met[s,dr,sc]
                               + demand_rejected[s,dr,sc];

subject to Ensure_Sub_Installed {k in SUBS,dr in DR_NODES,
                          n in PROC_NODES,sc in SCENARIOS}:
      demand_met_node[k,dr,n,sc] * (1 - ip_lp)
            <= sub_installed[k,n,sc] * large_number;

subject to Including_Routing {k in SUBS,dr in DR_NODES,
                          n in PROC_NODES,sc in SCENARIOS}:
      sum {r in ROUTES[dr,n]} routing[k,dr,n,r,sc] =
           demand_met_node[k,dr,n,sc] * flow_conversion[k];

subject to Services_Subservices {k in SUBS,dr in DR_NODES,
                                          sc in SCENARIOS}:
      sum {n in PROC_NODES} demand_met_node[k,dr,n,sc]
           = sum {ss in SUBSFOR[k]} service_use[k,ss] *
                                    demand_met[ss,dr,sc];

subject to Fixing_Subs_Vbles_Zero {k in SUBS,n in PROC_NODES
                                    ,sc in SCENARIOS}:
      sub_installed[k,n,sc] <= (1 - ip_lp);
```

C.3. PC1 Data File

base-frsdr.run

```
# samples a 'base-case' spread of demand, from which scenarios
# are found
```

```
# determining the size of each 'dr' location,
for {dr in DR_NODES} {

      let dr_base_random := Uniform01();

      for {ddr in 1..dr_num_sizes} {
            if dr_base_random > (ddr - 1) / dr_num_sizes and
            dr_base_random <= ddr / dr_num_sizes then
                  let basedr_type[dr] := ddr;
      }
}

# setting the 'base-case' fr_sdr, by evenly spreading over all
# services
let sum_dr_type := sum {s in SERVICES,dr in DR_NODES}
                                          basedr_type[dr];
for {s in SERVICES,dr in DR_NODES} {
      let base_frsdr[s,dr] := basedr_type[dr] / sum_dr_type;
}
```

get-frsdr.run

```
if num_scenarios >= 1 and num_scenarios <= 10 then {

      # determine which services are constant and which are potential
      # boom/fail over the horizon
      for {s in SERVICES} {
            let s_random := Uniform01();
            if s_random <= probab_s_con then {
                  # service is constant
                  let s_type[s] := 1;
            }
            if s_random > probab_s_con and s_random <=
            (probab_s_con + probab_s_small) then {
                  # service will change by a small amount
                  let s_type[s] := 2;
            }
            if s_random > (1 - probab_s_big) then {
                  # service will change by a big amount
```

```
                let s_type[s] := 3;
        }
    }

    # determine which demand locations are constant and which are
    # potential boom/fail over the horizon
    for {dr in DR_NODES} {
        let dr_random := Uniform01();
        if dr_random <= probab_dr_con then {
            # dr is constant
            let dr_type[dr] := 1;
        }
        if dr_random > probab_dr_con and dr_random <=
        (probab_dr_con + probab_dr_small) then {
            # dr will change by a small amount
            let dr_type[dr] := 2;
        }
        if dr_random > (1 - probab_dr_big) then {
            # dr will change by a big amount
            let dr_type[dr] := 3;
        }
    }
}

# scenarios start as base case, from which they will change,
# depending on how their demand location and service booms/fails
for {s in SERVICES,dr in DR_NODES,sc in SCENARIOS} {
    let fr_sdr[s,dr,sc] := base_frsdr[s,dr];
}

for {s in SERVICES} {
    # set how much demand for each service changes by, based on
    # whether it is known to be constant or a potential boom/fail
    if s_type[s] = 1 then {
        for {sc in SCENARIOS} {
            let s_change[s,sc] := 1;
        }
    }
```

```
        if s_type[s] = 2 then {
                for {sc in SCENARIOS} {
                        let s_dir_random := Uniform01();
                        if s_dir_random <= probab_s_boom then {
                                # service is booming
                                let s_change[s,sc] :=
                                        Uniform(1,s_change_small_ub);
                        }
                        else {
                                # service is failing
                                let s_data := Uniform(1,s_change_small_lb);
                                let s_change[s,sc] := 1 / s_data;
                        }
                }
        }
        if s_type[s] = 3 then {
                for {sc in SCENARIOS} {
                        let s_dir_random := Uniform01();
                        if s_dir_random <= probab_s_boom then {
                                # service is booming
                                let s_change[s,sc] :=
                                Uniform(s_change_small_ub,s_change_big_ub);
                        }
                        else {
                                # service is failing
                                let s_data := Uniform(s_change_small_lb,
                                                        s_change_big_lb);
                                let s_change[s,sc] := 1 / s_data;
                        }
                }
        }

        for {dr in DR_NODES,sc in SCENARIOS} {
                let fr_sdr[s,dr,sc] := fr_sdr[s,dr,sc] * s_change[s,sc];
        }
}

for {dr in DR_NODES} {
```

```
# set how much demand at each demand location changes by,
# based on whether it is known to be constant or a potential
# boom/fail
if dr_type[dr]= 1 then {
      for {sc in SCENARIOS} {
            let dr_change[dr,sc] := 1;
      }
}
if dr_type[dr]= 2 then {
      for {sc in SCENARIOS} {
            let dr_dir_random := Uniform01();
            if dr_dir_random <= probab_dr_boom then {
                  # dr is booming
                  let dr_change[dr,sc] :=
                        Uniform(1,dr_change_small_ub);
            }
            else {
                  # dr is failing
                  let dr_data := Uniform(1,dr_change_small_lb);
                  let dr_change[dr,sc] := 1 / dr_data;
            }
      }
}
if dr_type[dr]= 3 then {
      for {sc in SCENARIOS} {
            let dr_dir_random := Uniform01();
            if dr_dir_random <= probab_dr_boom then {
                  # dr is booming
                  let dr_change[dr,sc] :=
                  Uniform(dr_change_small_ub,dr_change_big_ub);
            }
            else {
                  # dr is failing
                  let dr_data := Uniform(dr_change_small_lb,
                                          dr_change_big_lb);
                  let dr_change[dr,sc] := 1 / dr_data;
            }
      }
```

```
        }
        for {s in SERVICES,sc in SCENARIOS} {
              let fr_sdr[s,dr,sc] :=
                    fr_sdr[s,dr,sc] * dr_change[dr,sc];
        }
}

# let some s/dr combination's demand change in a 'bursty' nature
let change_random := Uniform01();
if change_random <= 0.2 then let num_indiv_changes := 0;
if change_random > 0.2 and change_random <= 0.36 then
        let num_indiv_changes := 1;
if change_random > 0.36 and change_random <= 0.52 then
        let num_indiv_changes := 2;
if change_random > 0.52 and change_random <= 0.68 then
        let num_indiv_changes := 3;
if change_random > 0.68 and change_random <= 0.84 then
        let num_indiv_changes := 4;
if change_random > 0.84 and change_random <= 1 then
        let num_indiv_changes := 5;

for {ch in 1..num_indiv_changes,sc in SCENARIOS} {

        # a s/dr changes (booms/fails)
        let sdr_random := Uniform01();
        let sdr_dir_random := Uniform01();
        let number := 0;

        for {s in SERVICES,dr in DR_NODES} {

              let number := number + 1;

              if sdr_random > (number - 1) /
                                (num_drnodes * num_services)
              and sdr_random <= number / (num_drnodes * num_services)
              then {
                    if sdr_dir_random <= probab_sdr_boom then {
                          # s/dr is booming
```

```
                        let sdr_change := Uniform(1,sdr_change_ub);
                }
                else {
                        # s/dr is failing
                        let sdr_data := Uniform(1,sdr_change_lb);
                        let sdr_change := 1 / sdr_data;
                }
                let fr_sdr[s,dr,sc] :=
                        fr_sdr[s,dr,sc] * sdr_change;
        }
    }
}

for {sc in SCENARIOS} {

    # re-scale frsdr to sum to 1
    let sum_frsdr := sum {s in SERVICES,dr in DR_NODES}
                                        fr_sdr[s,dr,sc];

    for {s in SERVICES,dr in DR_NODES} {
        let fr_sdr[s,dr,sc] := fr_sdr[s,dr,sc] / sum_frsdr;
    }
}

for {sc in SCENARIOS} {

    # add slight random deviations (+/- 5%)
    for {s in SERVICES,dr in DR_NODES} {
        let random_dev := Uniform(0.95, 1.05);
        let fr_sdr[s,dr,sc] := random_dev * fr_sdr[s,dr,sc];
    }
}

# include base case as a scenario
for {s in SERVICES,dr in DR_NODES} {
    let fr_sdr[s,dr,1] := base_frsdr[s,dr];
}
```

pc1-data.run

```
# defining set 'SUBSFOR': what services use what subservices
let fr_newmember := 0;
let fr_newservice := 0.02;
for {k in SUBS} {
      let u12 := Uniform01();
      let u13 := Uniform01();
      if k = 1 then {
            repeat {
                  let u7 := Uniform01();
                  let new_member := round(u7 * (num_services - 1)+1);
            } until new_member <> k;
            if u12 >= fr_newmember then
                  let SUBSFOR[k] := {k};
            else {
                  if u13 >= fr_newservice then
                        let SUBSFOR[k] := {k,new_member};
                  else {
                        let num_services := num_services + 1;
                        let SUBSFOR[k] := {k,num_services};
                  }
            }
      }
      if k <> 1 and k <> num_subs then {
            repeat {
                  let u7 := Uniform01();
                  let new_member := round(u7*(num_services - 1)+1,0);
            } until new_member <> k and new_member <> (k - 1);
            if u12 >= fr_newmember then
                  let SUBSFOR[k] := {k-1,k};
            else {
                  if u13 >= fr_newservice then {
                        if new_member < k then
                              let SUBSFOR[k] := {new_member,k-1,k};
                        if new_member > k then
                              let SUBSFOR[k] := {k-1,k,new_member};
                  }
```

```
                else {
                        let num_services := num_services + 1;
                        let SUBSFOR[k] := {k-1,k,num_services};
                }
        }
    }
    if k = num_subs then {
        repeat {
                let u7 := Uniform01();
                let new_member := round(u7*(num_services - 1)+1,0);
        } until new_member <> num_services;
        if u12 >= fr_newmember then
                let SUBSFOR[k] := {num_services};
        else {
                if u13 >= fr_newservice then
                        let SUBSFOR[k] := {new_member,num_services};
                else {
                        let num_services := num_services + 1;
                        let SUBSFOR[k] :=
                                {num_services-1,num_services};
                }
        }
    }
}

# setting parameter values
let disc_factor := 0.05;
let large_number := 10000;
let {k in SUBS} sub_conversion[k] := 1;
let {k in SUBS} flow_conversion[k] := 1;
let {s in SERVICES,k in SUBS} service_use[k,s] := 1;
let {rn in ROUTING_NODES} node_route_cost[rn] := 0;
let {s in SERVICES,dr in DR_NODES} rej_cost[s,dr] := 50000;
let {n in PROC_NODES} node_cost[n] := 0;
let {s in SERVICES,dr in DR_NODES} price[s,dr] := 400;

# setting routes between locations
for {dr in DR_NODES,n in PROC_NODES} {
```

```
      let ROUTES[dr,n] := {1};
}
let num_1 := 1;
let num_2 := 1;
for {rn in ROUTING_NODES} {
      if num_2 < num_pns then
            let num_2 := num_2 + 1;
      else {
            if num_2 = num_pns then {
                  let num_1 := num_1 + 1;
                  let num_2 := num_1 + 1;
            }
      }
      let ROUTES_THRU_NODES[rn] :=
            {(num_1,num_2,1),(num_2,num_1,1)};
}

# setting the amount of processing capacity it takes to install each
# subservice
let probab1 := 0.3333;
let probab2 := 0.6666;
let installamt := Uniform01();
if installamt >= 0 and installamt <= probab1 then {
      let total_subinstall := 200;
}
if installamt >= probab1 and installamt <= probab2 then {
      let total_subinstall := 600;
}
if installamt >= probab2 and installamt <= 1 then {
      let total_subinstall := 1000;
}
let min_subsinstall := 0.2 * total_subinstall;
for {k in SUBS} {
      let subslevel[k] := Uniform01();
      let installedsub_usage[k] := subslevel[k] * total_subinstall;
      if installedsub_usage[k] < min_subsinstall then
            let installedsub_usage[k] := min_subsinstall;
}
```

```
let sum_subsinstall := sum {k in SUBS} installedsub_usage[k];
for {k in SUBS} {
      let installedsub_usage[k] := installedsub_usage[k] /
                        sum_subsinstall * total_subinstall;
}

# setting parameter values for processing and routing node
# characteristics
let {rn in ROUTING_NODES} npiece_ttime[rn] := 5;
let {n in PROC_NODES} npiece_ptime[n] := 5;
let r4a := Uniform(0.6,1.4);
# multiplicative scalar factor for processing nodes
let r4b := Uniform(-0.3,1);
# additive scalar factor for processing nodes
for {n in PROC_NODES} {
      let invest_limit_unit[n] := 100000;
      let min_add[n] := 0;
      let maint_cost[n] := 0;
      let rate_ptime[n,1] := 1 * r4a + r4b;
      let rate_ptime[n,2] := 1.3333 * r4a + r4b;
      let rate_ptime[n,3] := 2 * r4a + r4b;
      let rate_ptime[n,4] := 4 * r4a + r4b;
      let rate_ptime[n,5] := 1000 ;
      let ptime_bkpt_fr[n,1] := 0.2;
      let ptime_bkpt_fr[n,2] := 0.2 ;
      let ptime_bkpt_fr[n,3] := 0.2 ;
      let ptime_bkpt_fr[n,4] := 0.2 ;
      let ptime_bkpt_fr[n,5] := 0.2 ;
}
let {n in PROC_NODES} invest_fixed[n] := 1000000;
let rand_invest_vble := Uniform(100,3000);
let {n in PROC_NODES} invest_vble[n] := rand_invest_vble;
let r5a := Uniform(0.6,1.4);
# multiplicative scalar factor for routing nodes
let r5b := Uniform(-0.3,2);
# additive scalar factor for routing nodes
for {rn in ROUTING_NODES} {
      let rate_ttime[rn,1] := 1 * r5a + r5b;
```

```
        let rate_ttime[rn,2] := 1.3333 * r5a + r5b;
        let rate_ttime[rn,3] := 2 * r5a + r5b;
        let rate_ttime[rn,4] := 4 * r5a + r5b;
        let rate_ttime[rn,5] := 1000 ;
        let limit_ttime[rn,1] := 200;
        let limit_ttime[rn,2] := 400;
        let limit_ttime[rn,3] := 600;
        let limit_ttime[rn,4] := 800;
}
```

definitions.run

```
# defines parameters used in base-frsdr.run and get-frsdr.run files

param base_frsdr {SERVICES,DR_NODES} >= 0, <= 1;
param basedr_type {DR_NODES} >= 0;
# 1 - small, 2 - medium, 3 - big
param change_random >= 0, <= 1;
param dr_base_random >= 0, <= 1;
param dr_change {DR_NODES,MAX_SCENARIOS} >= 0;
param dr_change_big_lb >= 0, := 1.5;
param dr_change_big_ub >= 0, := 1.5;
param dr_change_small_lb >= 0, := 1.25;
param dr_change_small_ub >= 0, := 1.25;
param dr_data >= 0;
param dr_dir_random >= 0, <= 1;
param dr_num_sizes >= 0, := 3;
param dr_random >= 0, <= 1;
param dr_type {DR_NODES} >= 0;
# 1 - constant, 2 - small change, 3 - big change
param num_indiv_changes >= 0;
param number >= 0 default 0;
param probab_dr_big >= 0, <= 1, := 0.25;
param probab_dr_boom >= 0, <= 1, := 0.5;
param probab_dr_con >= 0, <= 1, := 0.5;
param probab_dr_small >= 0, <= 1, := 0.25;
param probab_s_big >= 0, <= 1, := 0.25;
param probab_s_boom >= 0, <= 1, := 0.5;
param probab_s_con >= 0, <= 1, := 0.5;
```

```
param probab_s_small >= 0, <= 1, := 0.25;
param probab_sdr_boom >= 0, <= 1, := 0.5;
param random_dev;
param s_change {SERVICES,MAX_SCENARIOS} >= 0;
param s_change_big_lb >= 0, := 4;
param s_change_big_ub >= 0, := 4;
param s_change_small_lb >= 0, := 2;
param s_change_small_ub >= 0, := 2;
param s_data >= 0;
param s_random >= 0, <= 1;
param s_dir_random >= 0, <= 1;
param s_type {SERVICES} >= 0;
param sdr_change >= 0;
param sdr_change_lb >= 0, := 5;
param sdr_change_ub >= 0, := 5;
param sdr_dir_random >= 0, <= 1;
param sdr_data >= 0;
param sdr_random >= 0, <= 1;
param sum_dr_type >= 0;
param sum_frsdr >= 0;

# defines parameters used in pc1-data.run file

param fr_newmember >= 0;
param fr_newservice >= 0;
param installamt >= 0;
param min_subsinstall >= 0;
param new_member >= 0;
param num_1 >= 0;
param num_2 >= 0;
param probab1 >= 0, <= 1;
param probab2 >= 0, <= 1;
param subslevel {MAX_SUBS} >= 0, <= 1;
param u7 >= 0, <= 1;
param u12 >= 0 , <= 1;
param u13 >= 0 , <= 1;
param r4a ;
param r5a ;
```

```
param r4b ;
param r5b ;
param rand_invest_vble >= 0;
param sum_subsinstall >= 0;
param total_subinstall >= 0;

# defines parameters used in pc1.run file

param Approx_pc {PROC_NODES} >= 0;
param Assess_elrp binary;
param Base_pc {PROC_NODES} >= 0;
param num_runs >= 0;
```

C.4. PC1 Run File

```
# pc1.run

option solver_msg 1;
option show_stats 0;
option cplex_options 'timing=1' 'timelimit=10000';    # in seconds
option solution_round 10;
option randseed 1471013;

model pc1.mod;

# network size
let num_pns := 5;
let num_drnodes := num_pns;
let num_rns := (num_pns * (num_pns - 1)) / 2;
let num_subs := 5;
let num_services := num_subs - 1;

# model approximations
let ip_lp := 1;
let nt_pn := 0;
let int_lin := 0;

problem pc1Model;
```

```
let cust_exp := 7.5;
let exp_rejs := 0;
let maxm_dpool := 10000;

let num_runs := 100;

for {aa in 1..num_runs} {

      commands pc1-data.run;

      for {bb in 1..2} {
      # find stochastic and deterministic processing capacity
      # solutions
            if bb = 1 then {
                  let num_scenarios := 7;
                  commands base-frsdr.run;
                  commands get-frsdr.run;
            }
            if bb = 2 then {
                  let num_scenarios := 1;
                  for {s in SERVICES,dr in DR_NODES} {
                        let fr_sdr[s,dr,1] := base_frsdr[s,dr];
                  }
            }
            let {sc in SCENARIOS}
                  probab_scen[sc] := 1 / num_scenarios;

            solve pc1Model;

            if bb = 1 then
                  let {n in PROC_NODES} Base_pc[n] := pc_add[n];
            if bb = 2 then
                  let {n in PROC_NODES} Approx_pc[n] := pc_add[n];
      }

      let Assess_elrp := 1;
      if Assess_elrp = 1 then {
```

```
            let num_scenarios := 50;
            let {sc in SCENARIOS}
                  probab_scen[sc] := 1 / num_scenarios;
            commands get-frsdr.run;

            for {cc in 1..2} {
                  if cc = 1 then
                  let {n in PROC_NODES} pc_levels[n] := Base_pc[n];
                  if cc = 2 then
                  let {n in PROC_NODES} pc_levels[n] := Approx_pc[n];

                  problem rpModel;
                  commands rp.run;
            }
      }
}

reset;
```

C.5. RP Model File

```
# rp.mod

                              problem rpModel;
# ================================================================

# this model is the lower level processing decisions model. It
# allows you to set a processing capacity arrangement and then the
# determine lower level periods actions and associated profit with
# that processing capacity level.

# only sets and parameters additional to that defined in pc1.mod

# Sets

set PERIODS := {1..num_periods};
# processing decisions periods in the strategic planning horizon
```

```
# Parameters

param pc_levels {PROC_NODES} >= 0;
# processing capacity

param pc_avail {PROC_NODES} binary ;
# 1, if processing capacity is installed at a node, 0 otherwise

param Total_dpoola {SCENARIOS};
# the total demand pool for all service/demand location combinations
# in the period (updated period to period)

param Total_Dpool_initial >= 0;
# the total demand pool in the first period, which is the same for
# each scenario

param Average_Dpool {SCENARIOS} >= 0;
# average demand pool over all periods.
# Used to compare with LD from PC1

param ext_factors {PERIODS} ;
# new customers due to external factors at end of period p.

param npiece_dchange >= 0;
param limit_dchange {1..npiece_dchange+1};
# the step-wise function of change to demand due to QoS

param fr_rejs_mean <= 0;
param fr_rejs_sd >= 0;
param fr_dts_mean {1..npiece_dchange};
param fr_dts_sd {1..npiece_dchange} >= 0;
# the mean and standard deviation of fractional change to demand
# based on QoS (i.e., the distribution). Losses can be incurred
# due to rejections and delivery times, but gains can only be
# obtained due to delivery times.

param fr_rejs {PERIODS} ;
```

```
param fr_dts {1..npiece_dchange,PERIODS};
# the actual fractional changes to the demand pool based on QoS,
# where this is based on the distribution for the fractional change
# amount (mean and standard deviation)

param av_dt_segment {SCENARIOS,PERIODS,np in 1..npiece_dchange}
                                                    >= 0, <= 1;
# 1, if the total delivery time is on np'th step, 0 otherwise

param total_dpool_change_qos {SCENARIOS,PERIODS} ;
# actual changes to the demand pool due to both QoS factors.

param pds_profit_disc {SCENARIOS,PERIODS};
param Total_Dpool {SCENARIOS,PERIODS};
param lr_profit {SCENARIOS};
param expected_lr_profit;
param av_dta {SCENARIOS,PERIODS} >= 0;
param ttc_routing;
# penalty of delivery times

# Decision Variables

var pds_level_profita {SCENARIOS};
var Demanda {SERVICES,DR_NODES,SCENARIOS} >= 0;
var demand_meta {SERVICES,DR_NODES,SCENARIOS} >= 0;
var demand_rejecteda {SERVICES,DR_NODES,SCENARIOS} >= 0;
var demand_met_nodea {SUBS,DR_NODES,PROC_NODES,SCENARIOS} >= 0;
var sub_installeda {SUBS,PROC_NODES,SCENARIOS} binary;
# integer element removed with ip_lp = 1
var routinga {SUBS,dr in DR_NODES,n in PROC_NODES,ROUTES[dr,n],
                                            SCENARIOS} >= 0;
var delivery_costsa {SCENARIOS} >= 0;
var total_pc_used_demanda {PROC_NODES,SCENARIOS} >= 0;
var total_pc_used_subinstalla {PROC_NODES,SCENARIOS} >= 0;
var total_pc_useda {PROC_NODES,SCENARIOS} >= 0;
var proc_costsa {SCENARIOS} >= 0;
var proc_timea {PROC_NODES,SCENARIOS} >= 0;
var delivery_timea {SCENARIOS} >= 0;
```

```
var transp_timea {SCENARIOS} >= 0 ;
var time_through_nodea {ROUTING_NODES,SCENARIOS} >= 0;
var dm_flow_nodea {ROUTING_NODES,SCENARIOS} >= 0;
var pc_segment_useda {n in PROC_NODES,1..npiece_ptime[n],SCENARIOS}
>= 0;
var pc_segment_used_subsa {n in
PROC_NODES,1..npiece_ptime[n],SCENARIOS} >= 0;
var pc_segment_used_dema {n in
PROC_NODES,1..npiece_ptime[n],SCENARIOS} >= 0;
var av_pc_subsa {PROC_NODES} >= 0;

var z1a {n in PROC_NODES,1..npiece_ptime[n],SCENARIOS} binary;
var z2a {n in PROC_NODES,1..npiece_ptime[n],SCENARIOS} binary;
# integer element removed with 2nd_stage_ints = 1

# Objective Function

maximize pds_period_profit_discounted:

      sum {sc in SCENARIOS} pds_level_profita[sc];

      # profit from the current processing decisions periods, given
      # set capacity

# Constraints

subject to Calculating_pds_level_profitsa {sc in SCENARIOS}:
      pds_level_profita[sc] = sum {s in SERVICES,dr in DR_NODES}
            (demand_meta[s,dr,sc] * price[s,dr]
            - demand_rejecteda[s,dr,sc] * rej_cost[s,dr])

      - proc_costsa[sc]

      - delivery_costsa[sc] ;

subject to Find_proc_costsa {sc in SCENARIOS}:
      proc_costsa[sc] = sum {n in PROC_NODES}
                        total_pc_useda[n,sc] * node_cost[n];
```

```
subject to Processing_Capacity_Constrainta {n in PROC_NODES,
                                        sc in SCENARIOS}:
      total_pc_useda[n,sc] <= pc_levels[n] ;

subject to Find_total_pc_useda {n in PROC_NODES,sc in SCENARIOS}:
      total_pc_useda[n,sc] = total_pc_used_demanda[n,sc]
                        + total_pc_used_subinstalla[n,sc] ;

subject to Find_pc_used_for_demanda {n in PROC_NODES,
                                        sc in SCENARIOS}:
      total_pc_used_demanda[n,sc] = sum {k in SUBS,dr in DR_NODES}
                  demand_met_nodea[k,dr,n,sc] * sub_conversion[k];

subject to Find_av_pc_subsa {n in PROC_NODES,sc in SCENARIOS}:
      av_pc_subsa[n] =
                  (1 - nt_pn) * (pc_avail[n] * coeff1_Nt *
                                          sum {k in SUBS}
                        installedsub_usage[k] / num_pns) +
                        nt_pn * (pc_avail[n] * coeff1_Pn
                                          * sum {k in SUBS}
                        installedsub_usage[k] / num_pns
                        + coeff2_Pn * pc_levels[n]);

subject to Find_pc_used_for_subinstalla {sc in SCENARIOS,
                                        n in PROC_NODES}:
      total_pc_used_subinstalla[n,sc] = av_pc_subsa[n] * ip_lp
            + (1 - ip_lp) *
            sum {k in SUBS} sub_installeda[k,n,sc]
                        * installedsub_usage[k];

subject to Finding_Pc_breakpointsa {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      pc_segment_useda[n,q,sc] <= ptime_bkpt_fr[n,q] * pc_levels[n];

subject to Finding_Pc_breakpoints_ba {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      pc_segment_useda[n,q,sc] =
```

```
      pc_segment_used_subsa[n,q,sc] + pc_segment_used_dema[n,q,sc];

subject to Summing_segmentsa {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 0}:
      sum {q in 1..npiece_ptime[n]}
            pc_segment_useda[n,q,sc] = total_pc_useda[n,sc];

subject to Summing_segments_subsa {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 0}:
      sum {q in 1..npiece_ptime[n]}
            pc_segment_used_subsa[n,q,sc] =
                  total_pc_used_subinstalla[n,sc];

subject to Summing_segments_demanda {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 0}:
      sum {q in 1..npiece_ptime[n]}
            pc_segment_used_dema[n,q,sc] =
                  total_pc_used_demanda[n,sc];

subject to Setting_dem_segments_1a {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      total_pc_used_subinstalla[n,sc]
            - ptime_bkpt_fr[n,q] * pc_levels[n]
            <= large_number * (1 - z1a[n,q,sc]);

subject to Setting_dem_segments_2a {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      pc_segment_used_dema[n,q,sc] <= large_number * z1a[n,q,sc];

subject to Setting_subs_segments_1a {n in PROC_NODES,
      q in 1..npiece_ptime[n],sc in SCENARIOS: 2nd_stage_ints = 0}:
      ptime_bkpt_fr[n,q] * pc_levels[n]
            - total_pc_used_subinstalla[n,sc]
            <= large_number * (1 - z2a[n,q,sc]);

subject to Setting_subs_segments_2a {n in PROC_NODES,sc in SCENARIOS,
      q in 1..npiece_ptime[n]:q < npiece_ptime[n] and
                                    2nd_stage_ints = 0}:
```

```
      pc_segment_used_subsa[n,q+1,sc] <= large_number * z2a[n,q,sc];

subject to Find_proc_timea {n in PROC_NODES,sc in SCENARIOS}:
      proc_timea[n,sc] = (sum {q in 1..npiece_ptime[n]}
                  rate_ptime[n,q] * pc_segment_useda[n,q,sc]
                   - sum {q in 1..npiece_ptime[n]}
                  rate_ptime[n,q] * pc_segment_used_subsa[n,q,sc])
            * (1 - 2nd_stage_ints)
                  + rate_ptime_linear * total_pc_used_demanda[n,sc]
            * 2nd_stage_ints;

subject to Effective_capa {n in PROC_NODES,sc in SCENARIOS
                              : 2nd_stage_ints = 1}:
      total_pc_useda[n,sc] <= (1 - ptime_bkpt_fr[n,5])
                                          * pc_levels[n];
# an 'effective capacity' in the case of linear processing time
# functions

subject to Find_dm_flow_nodea {rn in ROUTING_NODES,sc in SCENARIOS}:
      dm_flow_nodea[rn,sc]=
            sum {k in SUBS,dr in DR_NODES,n in PROC_NODES,
                        r in ROUTES[dr,n]:
                  (dr,n,r) in ROUTES_THRU_NODES[rn]}
                              routinga[k,dr,n,r,sc] ;

subject to Find_routing_timea {rn in ROUTING_NODES,sc in SCENARIOS}:
      time_through_nodea[rn,sc] =
      << {q in 1..npiece_ttime[rn]-1} limit_ttime[rn,q];
            {q in 1..npiece_ttime[rn]} rate_ttime[rn,q] >>
                        dm_flow_nodea[rn,sc];

subject to Find_transp_timea {sc in SCENARIOS}:
      transp_timea[sc] = sum {rn in ROUTING_NODES}
                              time_through_nodea[rn,sc];

subject to Find_delivery_timea {sc in SCENARIOS}:
      delivery_timea[sc] = sum {n in PROC_NODES}
                              proc_timea[n,sc] + transp_timea[sc] ;
```

```
subject to Find_delivery_costsa {sc in SCENARIOS}:
      delivery_costsa[sc] =
           sum {rn in ROUTING_NODES} node_route_cost[rn] *
              sum {k in SUBS, (ddr,nn,rr) in ROUTES_THRU_NODES[rn]}
                  routinga[k,ddr,nn,rr,sc]
            + delivery_timea[sc] * ttc_routing;

subject to Finding_indiv_demanda {s in SERVICES,dr in DR_NODES,
                                          sc in SCENARIOS}:
      Demanda[s,dr,sc] = fr_sdr[s,dr,sc] * Total_dpoola[sc] ;

subject to Meet_Demanda {s in SERVICES,dr in DR_NODES,
                                    sc in SCENARIOS}:
      Demanda[s,dr,sc] = demand_meta[s,dr,sc]
                              + demand_rejecteda[s,dr,sc];

subject to Ensure_Sub_Installeda {k in SUBS,dr in DR_NODES,
                          n in PROC_NODES,sc in SCENARIOS}:
      demand_met_nodea[k,dr,n,sc] * (1 - ip_lp)
             <= sub_installeda[k,n,sc] * large_number;

subject to Including_Routinga {k in SUBS,dr in DR_NODES,
                          n in PROC_NODES,sc in SCENARIOS}:
      sum {r in ROUTES[dr,n]} routinga[k,dr,n,r,sc] =
            demand_met_nodea[k,dr,n,sc] * flow_conversion[k];

subject to Services_Subservicesa {k in SUBS,dr in DR_NODES,
                                          sc in SCENARIOS}:
      sum {n in PROC_NODES} demand_met_nodea[k,dr,n,sc]
            = sum {ss in SUBSFOR[k]} service_use[k,ss] *
                                    demand_meta[ss,dr,sc];

subject to Fixing_Subs_Vbles_Zeroa {k in SUBS,n in PROC_NODES
                                    ,sc in SCENARIOS}:
      sub_installeda[k,n,sc] <= (1 - ip_lp);
```

C.6. RP Data File

```
# rp.dat - data file for rp.mod, remaining parameters are defined in
# run files

param ttc_routing := 1;

param npiece_dchange := 7 ;

param limit_dchange :=  1     -100000
                        2     -4
                        3     -1
                        4     -0.1
                        5     0.1
                        6     1
                        7     4
                        8     100000      ;

# 'low' rate of change
param fr_dts_mean :=    1     0.1
                        2     0.05
                        3     0.02
                        4     0
                        5     -0.02
                        6     -0.05
                        7     -0.1  ;

# 'high' rate of change
/*
param fr_dts_mean :=    1     0.2
                        2     0.1
                        3     0.05
                        4     0
                        5     -0.05
                        6     -0.1
                        7     -0.2  ;
*/
```

```
param fr_dts_sd :=      1     0.02
                        2     0.001
                        3     0.001
                        4     0
                        5     0.001
                        6     0.001
                        7     0.02  ;

param fr_rejs_mean := -0.5 ;
param fr_rejs_sd := 0.02 ;
```

C.7. RP Run File

```
# rp.run

# this model takes the processing capacity solution and then uses the
# RP model to estimate the expected long-run profit of that
# solution.

for {n in PROC_NODES} {
      if pc_levels[n] > 0 then
            let pc_avail[n] := 1;
      else
            let pc_avail[n] := 0;
}

for {sc in SCENARIOS} {

      let Total_dpoola[sc] := Total_Dpool_initial;

      for {p in PERIODS} {

            # for recording purposes
            let Total_Dpool[sc,p] := Total_dpoola[sc];

            # solve period model for period's demand
            solve rpModel;
```

```
# 'real' period profit
let pds_profit_disc[sc,p] :=
      pds_period_profit_discounted
            * disc_factor ^ (p-1)
            + delivery_timea[sc] * ttc_routing;

# average delivery time in meeting period's demand
if sum {s in SERVICES,dr in DR_NODES}
      demand_meta[s,dr,sc] = 0
then
      let av_dta[sc,p] := 0;
else {
      let av_dta[sc,p] := delivery_timea[sc] /
            sum {s in SERVICES,dr in DR_NODES}
                  demand_meta[s,dr,sc];
}

if p < num_periods then {

# the actual fractional change due to rejections and
# delivery times, based on the given distribution for
# these parameters
      let fr_rejs[p] :=
            Normal(fr_rejs_mean,fr_rejs_sd);

      for {np in 1..npiece_dchange} {
            let fr_dts[np,p] :=
                  Normal(fr_dts_mean[np],fr_dts_sd[np]);
      }

# by comparing the actual average delivery times with the
# delivery times customers expect the 'step' on the
# step-wise function is determined
      if sum {s in SERVICES,dr in DR_NODES}
            demand_meta[s,dr,sc] > 0 then {

            for {np in 1..npiece_dchange} {
                  if (delivery_timea[sc] /
```

```
                        sum {s in SERVICES,
                              dr in DR_NODES}
                        demand_meta[s,dr,sc]
                              - cust_exp)
                              > limit_dchange[np]
                  and (delivery_timea[sc] /
                        sum {s in SERVICES,
                              dr in DR_NODES}
                        demand_meta[s,dr,sc]
                              - cust_exp)
                              <= limit_dchange[np+1]
                  then
                        let av_dt_segment[sc,p,np] := 1;
                  else
                        let av_dt_segment[sc,p,np] := 0;
            }
            }
      else {
            for {np in 1..npiece_dchange} {
                  let av_dt_segment[sc,p,np] := 0;
            }
      }

# overall change to demand pool based on 'step' of step-
# wise function, determined above, and number of
# rejections.
      let total_dpool_change_qos[sc,p] :=
            sum {s in SERVICES,dr in DR_NODES}
            fr_rejs[p] * demand_rejecteda[s,dr,sc]
            + sum {np in 1..npiece_dchange}
            fr_dts[np,p] * av_dt_segment[sc,p,np]
            * Total_dpoola[sc];

# updates demand pool taking into account changes due
# to quality of service and external factors, and
# allowing for maximum/minimum demand pool levels
      if total_dpool_change_qos[sc,p]
                        <= -1 * Total_dpoola[sc] then
```

```
                        let Total_dpoola[sc] := 0;
                  else {
                        if total_dpool_change_qos[sc,p] >=
                        (maxm_dpool - Total_dpoola[sc]) then
                              let Total_dpoola[sc] := maxm_dpool;
                        else
                              if (Total_dpoola[sc]
                              + total_dpool_change_qos[sc,p]) < 0.1
                              and (Total_dpoola[sc]
                              + total_dpool_change_qos[sc,p]) > -0.1
                              then
                                    let Total_dpoola[sc] := 0;
                              else
                                    let Total_dpoola[sc] :=
                                    Total_dpoola[sc]
                                    + total_dpool_change_qos[sc,p];
                  }
                  let Total_dpoola[sc] := Total_dpoola[sc]
                                          + ext_factors[p];
                  if Total_dpoola[sc] < 0 then
                        let Total_dpoola[sc] := 0;
            }
      }
      # profit over all periods
      let lr_profit[sc] := sum {pp in PERIODS}
                        pds_profit_disc[sc,pp] - investment_costs;
}
# expected profit, considering different scenarios
let expected_lr_profit := sum {sc in SCENARIOS} probab_scen[sc] *
                              lr_profit[sc];

# average total demand pool
for {sc in SCENARIOS} {
      let Average_Dpool[sc] := sum {p in PERIODS}
                              Total_Dpool[sc,p] / num_periods;
}
```

C.8. Run File Comparing PC1 and RP models

```
# pc1vrp.run

# tests whether long-run demand (from PC1) is a reasonable
# approximation of average demand (from RP), even when external
# factors are > 0.

option solver_msg 1;
option cplex_options 'timing=1' 'timelimit=10000';     # in seconds
option solution_round 10;
option randseed 1471013;

model pc1.mod;
model rp.mod;
data rp.dat;
commands definitions.run;

# setting network size
let num_pns := 5;
let num_drnodes := num_pns;
let num_rns := (num_pns * (num_pns - 1)) / 2;
let num_subs := 5;
let num_services := num_subs - 1;
let num_periods := 50;

# setting model approximations (none in this instance)
let 2nd_stage_ints := 0;
let 1st_stage_ints := 0;
let ip_lp := 0;
let nt_pn := 0;
let num_scenarios := 7;
let {sc in SCENARIOS} probab_scen[sc] := 1 / num_scenarios;

# setting model parameters
let cust_exp := 7.5;
let exp_rejs := 0;
let maxm_dpool := 10000;
```

```
let num_runs := 200;

for {aa in 1..num_runs} {
      # data set for each run
      commands pc1-data.run;
      commands base-frsdr.run;

      # test under high and low demand volatility
      for {p in PERIODS} {
            # low demand volatility
            let ext_factors[p] := Uniform(-250,500);
                                    # +/- 5% mdp
            # high demand volatility
            #let ext_factors[p] := Uniform(-1000,2000);
                                    # +/- 20% mdp
      }
      # initial demand close to expected average demand (limiting
      # start-up effects)
      let Total_Dpool_initial := 10000;

      for {bb in 1..2} {
            if bb = 1 then {
                  # find processing capacity levels and long-run
                  # demand from PC1 model
                  problem pc1Model;
                  solve pc1Model > pc1vrp.res;
            }
            if bb = 2 then {
                  # find average demand from RP model for the given
                  # processing capacity levels
                  problem rpModel;
                  let {n in PROC_NODES} pc_levels[n] := pc_add[n];
                  commands rp.run;
            }
      }
}
reset;
```